Содержание

6 Программирование: Java, сеть и Web

Существует огромное количество книг и статей, посвященных Java, в которых эта технология описывается как язык, платформа и даже как средство разработки сценариев (имеется в виду технология JSP — Java Server Pages). В данной главе рассматривается несколько реализаций и стандартных расширений Java, которые могут оказаться чрезвычайно важными для обеспечения взаимодействия платформ Windows 2000, Windows XP и Red Hat Linux 7.1 в компании Elsolutions. Язык Java прославился благодаря переносимости бинарного исполняемого кода между множеством платформ. Разработчики технологии Java обещают нам: «пишешь один раз, запускаешь на любой платформе» (write once, run anywhere). Для крупных монолитных приложений это утверждение можно оспорить, однако оно оказывается справедливым, если речь идет о структурных рабочих средах, таких как сервер приложений Java. В подобных рабочих средах для создания единого масштабируемого web-приложения используются самые разнообразные технологии, механизмы и парадигмы — например, бинарные компоненты, сценарии, транзакции и постановка в очередь. Для обеспечения функционирования любой из этих составляющих вы можете использовать как Windows, так и Linux.

Термин «web-программирование» означает создание масштабируемого приложения, осуществляющего обработку бизнес-данных, и работающего на web-сервере или сервере приложений. Большая часть web-приложений компании Elsolutions — это сценарии интранет и Интернет, а также приложения Java.

В данной главе рассматриваются полезные программные интерфейсы, прикладные рабочие среды и инструменты, используемые для разработки приложений в среде Web. Кроме того, мы рассмотрим варианты взаимодействия Windows XP и Red Hat Linux 7.1 в сфере разработки web-приложений в компании Elsolutions.

Java

Java — это одновременно и язык, и платформа. Технология Java была разработана компанией Sun Microsystems. Язык Java обладает всем необходимым для разработки объектно-ориентированных приложений. На самом деле написать на Java программу, которая не была бы объектно-ориентированной, невозможно. Весь исполняемый исходный код Java располагается в рамках определений классов. Любые функции являются членами классов, то есть методами. Любая программа Java — это набор взаимодействующих между собой объектов. Однако некоторые элементы языка не являются объектами: например, объектами не являются некоторые примитивные типы данных, такие как int, long и Boolean. Возможно, разработчики Java пошли на этот компромисс для увеличения производительности программ Java. Для того чтобы работать с примитивными данными, как с объектом, используются классы-оболочки (для сравнения: язык Microsoft C#, являющийся конкурентом Java, не использует примитивных типов — все типы C# являются объектами).

Платформа Java реализована в виде виртуальной машины Java (Java Virtual Machine, JVM). Компилятор Java транслирует исходный код Java в исполняемый байт-код Java. Интерпретатор JVM исполняет байт-код. Тот факт, что байт-код исполняется в режиме интерпретации, означает, что в большинстве случаев приложение Java работает медленнее, чем аналогичное приложение, компилируемое в язык машинных команд. Однако в последнее время для исполнения приложений Java используются новые технологии оптимизации, основанные на методике JIT (Just-In-Time). В момент запуска программы Java компилятор JIT преобразует байт-код Java в исполняемый код машинных команд. Благодаря этому производительность приложений Java в некоторых случаях становится соизмеримой с производительностью обычных компилированных программ, написанных на таких языках, как C++ и C. Разработанную компанией Microsoft технологию .NET часто сравнивают с JVM. Основным отличием .NET является то, что байт-код .NET никогда не интерпретируется — он всегда компилируется в обычный исполняемый код машинных команд. Зачастую это происходит в момент первого запуска программы, основанной на .NET.

Именно платформа Java, адаптированная для многих аппаратно-программных архитектур, представляет интерес с точки зрения переносимости и взаимодействия разнообразных платформ. Чтобы выполнить в некоторой среде бинарный исполняемый файл Java, требуется установить в этой среде JVM и набор стандартных библиотечных пакетов. Обратите внимание, что вам не потребуется заново перекомпилировать программу Java. Один и тот же бинарный файл можно запустить на исполнение везде, где есть JVM, то есть, например, в Windows, Linux, Sun Solaris или Mac OS. Другими словами, один и тот же исполняемый код Java без перекомпиляции можно запустить на любом процессоре, в любой операционной системе. Это магическое свойство Java стало основным привлекательным фактором, благодаря которому на Java обратили внимание фактически все крупные и мелкие игроки компьютерной индустрии. Возникло огромное множество стандартных библиотек и программных интерфейсов.

С самого момента появления на свет язык Java был сетевым языком. Встроенная в Java стандартная библиотека работы с сетью позволяет создавать распределенные сетевые приложения, основанные на распространенных низкоуровневых сетевых протоколах. Высокоуровневая распределенная обработка данных основана на таких протоколах, как RMI (Remote Method Invocation). Благодаря поддержке протокола CORBA Internet Inter-ORB Protocol (IIOP) реализуется распределенная обработка данных с участием сетевых узлов, не являющихся узлами Java.

Java 2™

Существует три основных архитектурных модификации Java. В начале на свет появилась версия Java 1.0x. Затем в результате переработки архитектуры появилась версия Java 1.1x; при этом некоторые из использовавшихся ранее программных интерфейсов были признаны неудачными. В версии Java 1.2 архитектура платформы была дополнительно улучшена. Эта последняя версия распространяется под торговой маркой Java 2.

ПРИМЕЧАНИЕ

Комплект разработчика Java 2 SDK версии 1.2.2 не будет работать в среде Red Hat Linux 7.1. Те, кто желает использовать Java 2 SDK в среде Red Hat 7, могут попробовать поработать с новой версией Java 2 SDK v 1.3.0 for Linux. Пакет Java 2 SDK v1.3.0 тестировался в системе Red Hat 6.1, официально он не поддерживается в Red Hat 7, однако при его использовании в этой ОС никаких проблем не замечено, если только не использовать классическую виртуальную машину (Classic VM).

Под общим названием Java 2 в свет выпущены три разновидности Java, предназначенных для разработки решений на разных уровнях. Варианты Java 2 перечислены в табл. 6.1.

Таблица 6.1. Различные редакции платформы Java 2

Редакция	Сокращенное обозначение	Целевая платформа
Java 2 Micro Edition (редакция микро)	J2ME	«Внедренные» (embedded) приложения
Java 2 Standard Edition (стандартная редакция)	J2SE	Клиент/рабочая станция
Java 2 Enterprise Edition (редакция для предприятий)	J2EE	Сервер приложений Java

Java 2 Standard Edition

В состав последних версий продукта J2SE, распространяемого компанией Sun, входят:

- Java 2 SDK, Standard Edition, v 1.3 (JDK) — комплект разработки приложений;
- Java 2 Runtime Environment, Standard Edition, v 1.3 (JRE) — среда исполнения приложений;
- Java 2 Platform, Standard Edition, v 1.3 Documentation (Docs) — документация.

С этим связаны следующие технологии Java:

- Java Plug-in;
- Java HotSpot Server Virtual Machine (виртуальная машина);
- Java 2 Platform, Standard Edition source code (исходный код).

На рис. 6.1. показана структура Sun J2SE.

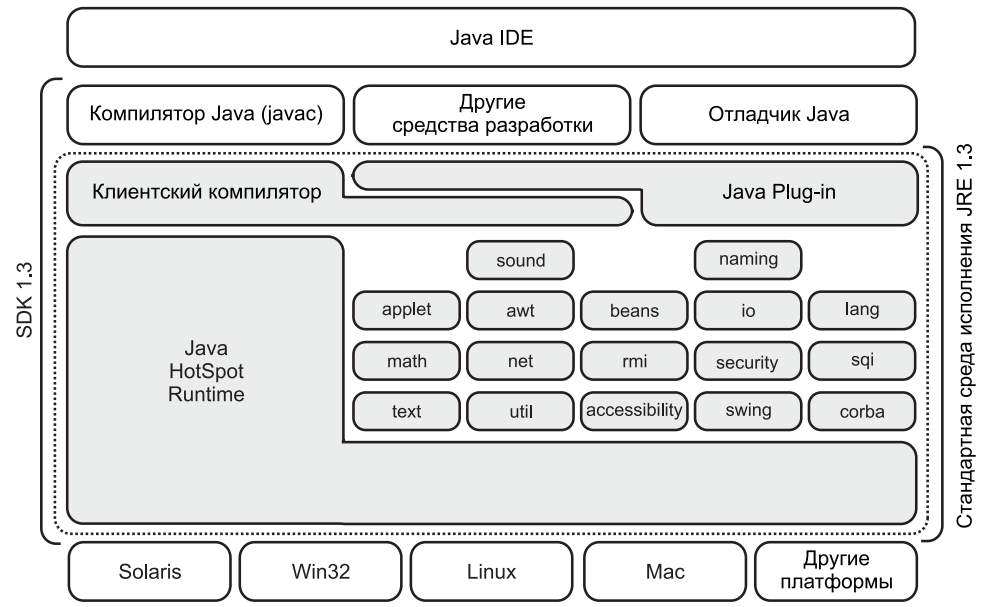

Рис. 6.1. Структура Sun Java 2 SDK, Standard Edition v 1.3

Комплект Java 1.3 SDK для Linux поставляется в двух формах: пакет RPM размером около 25 Мбайт и архив tar размером 27 Мбайт. И то и другое можно загрузить из Интернета в виде нескольких файлов, каждый из которых может уместиться на гибкий диск. Чтобы выполнить установку Java SDK, вы должны загрузить все без исключения части пакета.

Java 2 Enterprise Edition

Продукт Sun J2EE обеспечивает инфраструктуру для создания серверов приложений Java. Сервер приложения можно сертифицировать на совместимость со

стандартом J2EE. Совместимость с этим стандартом означает, что сервер поддерживает несколько технологий программирования, перечисленных в табл. 6.2.

Таблица 6.2. Программные интерфейсы Java Enterprise

Интерфейс	Описание
Enterprise Java Beans (EJB)	Модель компонентов для реализации бизнес-данных и бизнес-логики
Java Interface Description Language (IDL)	Независимое от языка описание распределенных объектов — устаревший метод доступа
Java Server Pages (JSP)	Код Java, встроенный в HTML-документ, компилируемый в виде сервлета
JDBC	Независимый от реализации конкретной базы данных интерфейс доступа к системе управления базой данных (RDBMS Java API)
Java Naming and Directory Interface (JNDI)	Доступ к службам имен и службам каталога
Java Messaging Service (JMS)	Независимый от конкретной реализации интерфейс для доступа к службам сообщений
Java Transaction Service (JTS)	Управление распределенными транзакциями
Servlet	Работающий на стороне сервера код Java, предлагающий обслуживание для клиентов
Remote Method Invocation (RMI)	Простой механизм обращения к удаленным методам Java

Сервлеты Java, в особенности сервлеты HTTP, — это набор функционирующих в одно и то же время экземпляров программ, которые можно использовать многократно. Технология сервлетов Java напоминает технологию CGI, однако обеспечивает значительно более высокую производительность. Сервлеты обладают возможностью управления сеансами обслуживания клиентов, благодаря чему они позволяют сохранять информацию о состоянии сеанса обслуживания клиентов, использующих HTTP, который традиционно не позволяет сохранять информацию о состоянии.

Страницы JSP (Java Server Pages) — это документы HTML, которые содержат в себе сценарии, написанные на языке Java. Компиляцию страницы JSP выполняет сервер приложений; это происходит в момент, когда страница размещается на сервере, или в момент, когда происходит первое обращение к странице. В результате создается сервлет, который используется в дальнейшем при обращении клиентов к данной странице. Другой сервлет, как правило, использует JSP для динамического представления данных. Код JSP использует JavaBean для передачи данных сервлету и для получения данных от сервлета. Сервлет, не выполняющий презентацию, является контроллером, в то время как сервлет JSP играет роль представления в традиционной схеме «модель — представление — контроллер» (Model — View — Controller, MVC). Код модели может размещаться в прикладных компонентах JavaBeans или EJB. Для доступа к данным может ис-

пользоваться система управления базой данных (DBMS) или очередь сообщений.

EJB — это компоненты повторного использования, взаимодействие с которыми осуществляется через интерфейсы. Обращение к компонентам EJB осуществляется с использованием диспетчера компонентов, который называется контейнером EJB. Для обнаружения каждого установленного компонента EJB используется служба имен, которая является провайдером JNDI (например, каталог LDAP). Контейнер EJB управляет накоплением откомпилированных компонентов, а также компиляцией компонентов с использованием JIT. Благодаря этому большое количество клиентов получает возможность обращаться к меньшему количеству экземпляров компонентов. Таким образом достигается значительная экономия ресурсов системы. Масштабируемость достигается благодаря совместному использованию компонентов, а также благодаря распределению нагрузки в кластере. Надежность системы повышается за счет кластерного переключения в случае отказа (cluster failover).

Чтобы обратиться к интерфейсам EJB, клиентский сервлет использует RMI через IIOP. Как правило, клиентом EJB является сервлет, однако это вовсе не обязательно. К интерфейсу EJB может обратиться любой клиент, даже не являющийся приложением Java. Достаточно, чтобы клиент использовал для доступа к EJB протокол IIOP. Например, к интерфейсу EJB может обратиться удаленный клиент CORBA, написанный на языке C, — для этого он должен обратиться к локальному вызову-заглушке, сгенерированному в результате компиляции определения интерфейса EJB.

Существует две основных разновидности EJB: сеансовый EJB (session EJB) и элементный EJB (entity EJB). Сеансовый EJB, как правило, используется для хранения бизнес-логики, а элементный EJB служит для представления постоянных данных. Последняя версия спецификации J2EE включает в себя возможность использования очереди сообщений в качестве клиента EJB. Ожидается, что в то время как эта книга выйдет из печати, спецификация J2EE уже будет реализована в конкретных программных продуктах.

Существуют серверы приложений J2EE, предназначенные для работы в средах Linux, Windows 2000 и Windows NT. Когда появится сервер Windows XP, эти серверы будут поддерживать и его. Например, сервер приложений IBM WebSphere может использоваться на любой из этих платформ, а также на некоторых других платформах.

Реализация Apache Tomcat JSP поддерживает спецификацию J2EE лишь отчасти. Однако Apache Tomcat — это полноценный сервер приложений JSP и сервлетов Java, и мы можем использовать его в качестве одного из серверов приложений компании Elsolutions. Сервер Tomcat написан на Java, поэтому один и тот же бинарный дистрибутив будет с одинаковым успехом функционировать как в среде Windows, так и в среде Linux.

Реализации JVM

В настоящее время существует достаточно большое количество реализаций виртуальной машины Java (Java Virtual Machine, JVM). Kaffe Open VM — это реа-

лизация JVM с открытым исходным кодом, к которой прилагаются компилятор, библиотека классов, совместимая с Java 1.1, а также стандартные библиотеки, реализованные для нескольких целевых платформ. Виртуальная машина Kaffe реализована для 30 операционных систем, работающих на восьми типах процессоров. Kaffe является единственной реализацией JVM, поддерживающей технологии обеих конкурирующих компаний: Sun и Microsoft. Ранее реализация Kaffe развивалась в рамках проекта Jolt, целью которого являлась разработка полностью свободной реализации системы Java. В настоящее время распространением Kaffe занимается компания Transvisual Technologies Inc. В рамках Kaffe унифицированы как настольные, так и «внедренные» (embedded) реализации Java. Реализация Kaffe является абсолютно чистой с лицензионной точки зрения, так как написана целиком и полностью с нуля. Таким образом, система Kaffe свободна от любых лицензионных ограничений и отчислений. Система Kaffe распространяется на условиях публичной лицензии GPL (GNU Public License). Если не считать нескольких нереализованных элементов, виртуальная машина Kaffe полностью совместима со спецификацией Java 1.1; кроме того, в ней реализованы некоторые возможности спецификации Java 2. Разработка Kaffe продолжается усилиями множества добровольцев по всему земному шару в обычном режиме разработки программных продуктов с открытым исходным кодом. Виртуальная машина Kaffe поставляется в составе дистрибутивов Red Hat Linux и Debian Linux.

На текущий момент Kaffe JVM обладает следующими недостатками:

- отсутствует замена RMI;
- JIT компилирует любой код, даже если код исполняется только один раз;
- компилятор JIT не выполняет оптимизацию;
- библиотеки времени исполнения не являются заранее откомпилированными;
- отсутствует система безопасности Java (в частности, не хватает механизмов подписывания кода и инспекции стека; механизм верификации байт-кода не завершен; не работают модификаторы открытого/закрытого доступа);
- механизм сбора мусора не оптимизирован, однако интерфейс определен достаточно хорошо, благодаря чему в качестве механизма сбора мусора можно использовать свой собственный программный модуль.

Систему Kaffe можно загрузить либо в виде файла RPM, либо в виде пакета tarball. Кроме того, Kaffe поставляется в составе дистрибутива Red Hat Linux. Версия 1.0.6 была выпущена 25 июля 2000 года. Размер составляет около 3,42 Мбайт. Более подробную информацию можно получить по адресам http://www.kaffe.org/ и http://www.transvirtual.com/.

Компилятор Java разрабатывается в рамках проекта GNU и называется GCJ (GNU Compiler for Java). Этот компилятор входит в так называемую коллекцию компиляторов GNU (GNU Compiler Collection, GCC). Компилятор GCJ позволяет компилировать:

- исходный код Java в исполняемый код процессорных команд;
- исходный код Java в байт-код Java, хранящийся в файлах с суффиксом .class;

- байт-код Java в исполняемый код процессорных команд.

Проект GCJ включает в себя смесь компилируемого и интерпретируемого кода Java. Результат работы компилятора GCJ должен быть скомпонован с библиотекой libgcj, которая включает в себя:

- системные библиотеки классов;
- сборщик мусора;
- абстракцию системных программных потоков;
- интерпретатор байт-кода.

Для отладки целевых приложений, откомпилированных с использованием GCJ, используется стандартный отладчик GNU. Компилятор GCJ поддерживает большую часть возможностей Java 1.1, включая внутренние классы (inner classes). Отсутствует пакет RMI.

GCJ — это проект компании Cygnus. В 2000 году компания Cygnus стала собственностью компании Red Hat. Более подробную информацию можно получить по адресу http://sources.redhat.com/java/index.html. Чтобы загрузить и установить компилятор и библиотеки, следуйте опубликованным по этим адресам инструкциям. Теперь рассмотрим пару реализаций Java для среды Windows.

Реализации Java для Windows

Бесплатные реализации Java для Windows, а также среды разработки, основанные на инструментах командной строки, распространяются компаниями Sun и IBM. Когда технология Java только появилась на сцене, компания Microsoft расширила стандарт Sun Java для того, чтобы обеспечить взаимодействие Java и COM. Технология COM принадлежит Microsoft и используется только в среде Windows. Обе технологии хорошо соответствовали друг другу, так как и в Java, и в COM поддерживалась концепция программного интерфейса. Некоторые утверждали, что компоненты COM проще написать на Microsoft Java, а не на C или C++.

Проблема состояла в том, что такие компоненты могли быть использованы только в среде Windows. Кроме того, в реализации Microsoft Java отсутствовал механизм Java RMI, так как он конкурировал с принадлежащей Microsoft технологией DCOM (Distributed Component Object Model). DCOM — это протокол взаимодействия между удаленными компонентами COM через сеть.

Компания Sun обвинила компанию Microsoft в нарушении лицензионного соглашения. В результате судебного разбирательства компания Microsoft прекратила работу над собственной реализацией Java и перестала поставлять Java в составе новых версий своих операционных систем. Компания Microsoft официально заявила о том, что она не будет поставлять Java в составе Windows XP. Таким образом, реализация Microsoft Java не соответствует требованиям последних версий стандарта Java. Реализацию Microsoft Java не рекомендуется использовать для проектов взаимодействия Linux/Windows.

Реализации Java для Linux

Теперь рассмотрим некоторые реализации Java для Linux. Некоторые из них доступны также в среде Windows. Компания Sun предлагает реализацию J2SE для Linux, которая полностью совместима с реализацией J2SE для Windows. Комплект разработчика Sun JDK 1.3 работает в среде Red Hat Linux 7.1 без каких-либо замеченных проблем.

Виртуальная машина IBM JVM использует высокопроизводительный компилятор JIT, который генерирует приложения, по скорости выполнения зачастую соизмеримые с программами, написанными на C и C++. Эквивалентная реализация IBM JVM доступна для платформы Windows. Для этого продукта свойственна высокая степень портируемости и совместимости с Windows. Виртуальная машина IBM JVM вполне удовлетворительно работала на наших тестовых машинах Red Hat Linux 7.1. Компания Intel утверждает, что IBM JVM является одной из самых быстрых виртуальных машин Java, ориентированных на платформу Intel.

Kaffe является виртуальной машиной Java от компании Transvisual. Продукт Kaffe разработан с нуля, без использования исходного кода Sun, поэтому он чист с точки зрения лицензирования. Так как Kaffe является продуктом с открытым исходным кодом, его можно откомпилировать в любой среде, в которой доступен компилятор GNU C. Продукт Kaffe в основном используется в системах UNIX и Linux, однако существуют версии, откомпилированные для среды Windows. Эти версии созданы с использованием инструментов разработки GNU для Windows. Продукт Kaffe в основном поддерживает стандарт Java 1.1, однако он также частично совместим с Java 2.

В компании Elsolutions планируется использовать несколько разных JVM. Зачем нужно использовать несколько виртуальных машин Java? Разнообразные продукты, например web-браузеры, а также продукты и инструменты Java компании IBM, содержат в своем составе собственную индивидуальную реализацию JVM для управления рабочей средой и исполнением программ. Таким образом мы планируем, что на некоторых машинах Red Hat Linux 7.1 будет параллельно использоваться несколько разных JVM.

Инструменты Java

Как уже было сказано ранее, компания Sun поставляет комплект разработки приложений JDK (Java Development Kit). Компания IBM также поставляет базовую среду разработки приложений Java для Windows, Linux и нескольких других платформ. Виртуальная машина Java, разработанная компанией IBM, обладает чрезвычайно высокой производительностью. Для увеличения скорости исполнения программ обе компании используют технологию JIT.

Проект Jikes объединяет усилия многих разработчиков. Целью проекта Jikes является создание компилятора Java, позволяющего компилировать исходный код Java в байт-код, который полностью соответствует стандарту, обладает высокой надежностью, полным набором возможностей, высоким качеством; кроме

того, этот компилятор будет доступен бесплатно. Работа над проектом Jikes началась в исследовательском подразделении компании IBM (IBM Research Division). Компания IBM сделала исходный код доступным для широкой интернет-общественности; таким образом, Jikes стал первым проектом компании IBM с открытым исходным кодом. Этот проект управляется совместно несколькими компаниями и отдельными добровольцами. В процессе работы над компилятором, а также над сопутствующим программным обеспечением и документацией активно используются Интернет и Web.

Отличительными особенностями Jikes являются:

- невероятно высокая скорость компиляции;
- анализ зависимостей для компиляции по частям;
- строгое соответствие спецификации языка Java.

Проект Jikes лицензируется в соответствии с публичной лицензией компании IBM (IBM Public License), которая была одобрена организацией Open Source Initiative. В настоящее время в состав Jikes входят компилятор Jikes, генератор грамматического разбора Jikes (Jikes Parser Generator) и набор тестирования Jikes (Jikes Test Suite). Пакет Jikes включается в состав дистрибутивов Debian Linux, Free BSD, Red Hat Linux и Linux-Mandrake. Можно получить также бинарные файлы Jikes для среды Windows. Более подробная информация содержится по адресу http://www10.software.ibm.com/developerworks/opensource/jikes/project/.

Компания Sun поставляет графическую интегрированную среду разработки (IDE) приложений J2SE под названием Forte for Java. Редакцию Community Edition можно получить бесплатно в виде одного файла или в виде набора нескольких менее крупных файлов. Для платформы Linux общий размер пакета составляет 10,62 Мбайт, а для платформы Windows размер пакета составляет 12,60 Мбайт.

Продукт Forte целиком и полностью написан на Java. Он обладает открытой архитектурой, поддерживающей добавление встраиваемых программных модулей, разработанных партнерами Sun или сторонними производителями. Среда Forte включает в себя утилиты, мастера и шаблоны, упрощающие разработку приложений Java.

Компания Sun также продает редакцию Internet Edition, которая упрощает разработку и использование сервлетов J2EE и соединений с базами данных при помощи JDBC. Наиболее совершенной разновидностью Forte является редакция Enterprise Edition, которая поддерживает совместную разработку и интеграцию с сервером iPlanet.

Графическая среда Forte использует парадигму множественных вторичных перекрывающихся окон; внешний вид рабочего окна программы напоминает ранние версии Microsoft Visual Basic (рис. 6.2). Фактически существует пять рабочих пространств:

- Editing (редактирование);
- GUI Editiong (редактирование GUI);
- Browse (просмотр);
- Running (запуск);

- Debugging (отладка).

Выбор рабочего пространства осуществляется при помощи вкладок, расположенных в левом верхнем углу рабочего окна.

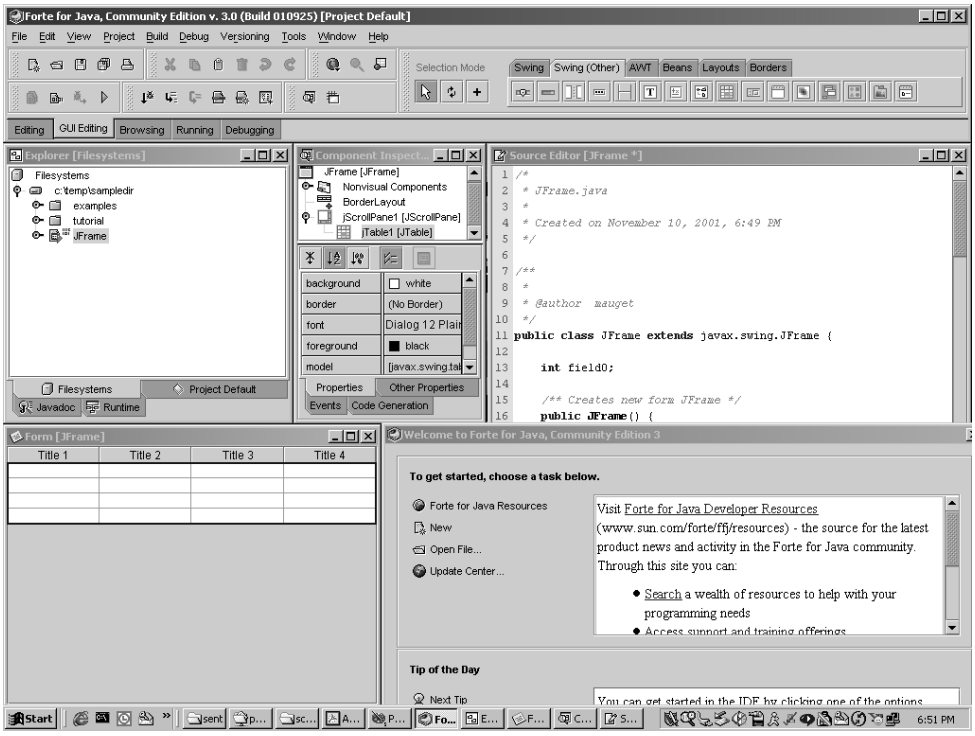

Рис. 6.2. Рабочее окно среды разработки Forte for Java

Продукт Forte можно запустить в любой версии Linux или Windows, благодаря чему Forte является превосходным переносимым средством разработки.

Альтернативой Forte является среда разработки IBM Visual Age for Java, которая часто обозначается сокращением VAJ. Существует три варианта VAJ: Entry Edition (начальная редакция), Professional Edition (профессиональная редакция) и Enterprise Edition (редакция для предприятий). VAJ Entry Edition можно получить бесплатно как для Linux, так и для Windows, однако в Linux этот инструмент позволяет создавать не более 500 классов, а в Windows — не более 750. В самом ближайшем времени компания IBM собирается выпустить редакции Professional и Enterprise для среды Linux. Возможно, в то время как вы читаете эту книгу, код уже доступен для использования.

Как и Forte, VAJ использует множество вторичных окон. Рабочее окно VAJ Enterprise Edition показано на рис. 6.3. Механизм распределения хранит каждую из версий и редакций кода в специальном хранилище. Рабочее окно VAJ отображает иерархию текущих редакций и версий кода и классов.

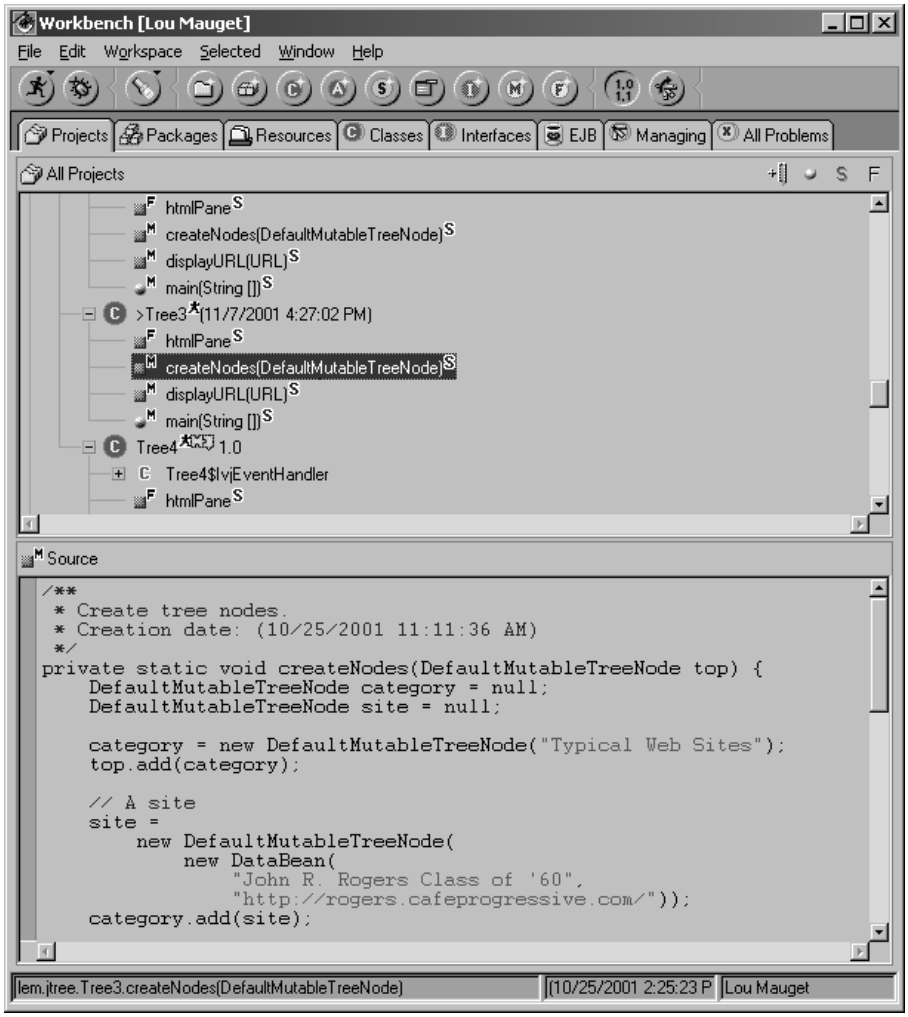

Рис. 6.3. Рабочее окно среды разработки IBM Visual Age for Java 3.53

В хранилище содержатся также все предыдущие версии и редакции. В рабочем пространстве отображается текущая редакция каждого из элементов приложения. В любой момент времени вы можете сделать текущей любую другую редакцию того или иного элемента. Формируя приложение, вы можете воспользоваться любой из версий того или иного пакета или класса. Работая со средой VAJ, чрезвычайно сложно потерять какой-либо код, так как в хранилище содержаться предыдущие версии *абсолютно всего*.

VAJ поддерживает свою собственную виртуальную машину и свой собственный сервер приложений для тестирования кода. Работая с VAJ, разработчик фактически никогда не сталкивается с необходимостью экспортировать класс или модуль исходного кода Java из хранилища в файловую систему. Экспорт

осуществляется только тогда, когда разрабатываемое приложение готово к использованию в реальной рабочей среде. Исходный код, хранящийся в хранилище, можно напрямую экспортировать в некоторые другие инструменты и рабочие среды IBM. Целевыми элементами при экспорте и импорте являются файлы, архивы Java (Java Archives, JAR) или файлы хранилища VAJ.

Редактор визуальных композиций позволяет разработчику графически соединять события, свойства и методы. Это весьма полезно при разработке апплетов и графических приложений Java. Элементы графического интерфейса размещаются на композиционную поверхность, а затем визуально соединяются с обработчиками и бизнес-логикой, реализованной в виде компонентов JavaBeans. Аналогичный композиционный редактор был реализован ранее в составе продукта IBM Visual Age for Smalltalk, а позже — в составе IBM Visual Age for C++.

Среда разработки VAJ обладает следующими преимуществами:

- проста в освоении и использовании;
- создает код для любой целевой платформы;
- поддерживает как Windows, так и Linux;
- поддерживает средства совместной разработки для команды разработчиков;
- безопасность версий обеспечена благодаря применению хранилища версий.

Более подробная информация располагается по адресу http://www-4.ibm.com/software/ad/vajava. Мы будем использовать VAJ for Java для работы над контрактами, а также для разработки собственного программного продукта Elsolutions.

Внедрение приложений Java

Приложение Java может быть реализовано в виде иерархии каталогов, которая соответствует иерархии в пакетном файле Java. Однако такое приложение достаточно сложно устанавливать и сопровождать; кроме того, файлы классов в этом случае не сжимаются. Из-за этого увеличивается время передачи исполняемого кода через сеть, а это замедляет работу распределенных приложений, использующих, например, апплеты Java. Загрузчик классов Java позволяет загружать классы из архивных файлов ZIP (технология сжатия ZIP широко распространена в среде Windows). Для этого файл ZIP должен быть добавлен в переменную окружения CLASSPATH. Именно так загружались библиотечные классы Java в среде Java 1.0. В настоящее время приложения Java, как правило, развертываются в виде пакетов классов JAR.

Файловый формат JAR (Java Archive), а также предназначенная для работы с ним команда jar впервые появились в составе Java 1.1. Формат команды jar совпадает с форматом команды tar, традиционно используемой в операционной системе UNIX. Основным отличием между двумя этими командами является используемый механизм сжатия файлов. Архив JAR содержит в себе файлы, сжатые в формате ZIP, используемом в Windows. Это отличный пример скрещивания двух разных технологий из Windows и UNIX. Для просмотра, обновления

и извлечения файлов из архива JAR можно использовать распространенное в среде Windows приложение WinZIP.

Классы Java загружаются по мере необходимости, при этом вне зависимости от платформы загрузчик классов просматривает каталоги, перечисленные в переменной окружения CLASSPATH. Формат этой переменной совпадает с форматом переменной PATH, однако, в отличие от переменной PATH, в составе переменной CLASSPATH можно указывать имена архивных файлов ZIP или JAR, — в этом случае загрузчик классов Java будет просматривать файловые иерархии, содержащиеся в этих архивах. Помимо переменной CLASSPATH, информацию о местоположении классов можно передать в командной строке команды java при помощи ключа –cp.

Развертывание приложений Java в среде J2EE осуществляется более сложно и разнообразно. Поставщики различных серверов приложений используют в своих продуктах различные методики и команды развертывания приложений Java. Общая особенность заключается в том, что сервлеты, сценарии JSP и компоненты EJB добавляются в работающий сервер, не нарушая его функционирования. Для этого используется процедура, которая может быть разной в разных продуктах. Сервлеты и сценарии JSP делятся на группы в соответствии с web-приложениями, в которых они используются. Иногда для обозначения этих групп используется термин «*зона сервлетов*» (servlet zone). Сложный сервер приложений может поддерживать распределение нагрузки (load balancing) и переключение в случае отказа (failover) различных web-приложений между серверами — членами кластера.

Компоненты EJB добавляются в состав контейнера EJB, являющегося частью сервера приложений. Это достаточно сложная процедура, так как на самом деле компонент EJB состоит из нескольких файлов, каждый из которых должен стать доступным для контейнера. Как правило, для обмена описательной информацией используется XML. Для того чтобы установить местоположение компонента EJB, используется служба имен, поэтому процедура установки компонента EJB предусматривает регистрацию местоположения и описания EJB при помощи провайдера JNDI, используемого сервером приложений.

Переносимость бинарного кода Java

Основной привлекательной и наиболее яркой отличительной чертой Java является переносимость бинарного кода. В некоторых ситуациях эта возможность слишком переоценивается. К сожалению, полной совместимости между различными реализациями Java удается достигнуть далеко не всегда. Многие реализации и приложения Java слишком сильно связаны с особенностями той или иной программно-аппаратной платформы. Обеспечить переносимость бинарного кода проще в случае если вы используете сервер J2EE. При этом вы должны следить за тем, чтобы ваше приложение не использовало функций, характерных только для конкретной реализации Java. Если это не так, значит, вы должны использовать одну и ту же реализацию Java на всех ваших платформах.

Пример: простой сервер обмена сообщениями, написанный на Java

Рассмотрим простое сетевое клиент-серверное приложение Java, которое можно было бы запускать как в Linux, так и в Windows. Написать такую программу достаточно просто. Обратите внимание на рис. 6.4. Здесь демонстрируется работа сервера обмена сообщениями (Chat Room Server), который работает на компьютере Linux и доступ к которому осуществляется при помощи клиента telnet, запущенного в среде Windows. Вы можете без проблем запустить те же самые бинарные файлы Java в среде Windows. Работа бинарного кода Java в среде Linux показана на рис. 6.5, а рис. 6.6 демонстрирует работу этого же самого кода в среде Windows. Обратите внимание, что версия Linux выдает в конце каждой строки некий артефакт. Это символ возврата каретки, который используется в среде Windows, однако игнорируется средой Linux. В каждом из случаев работает один и тот же бинарный файл, загруженный из сетевой папки общего доступа.

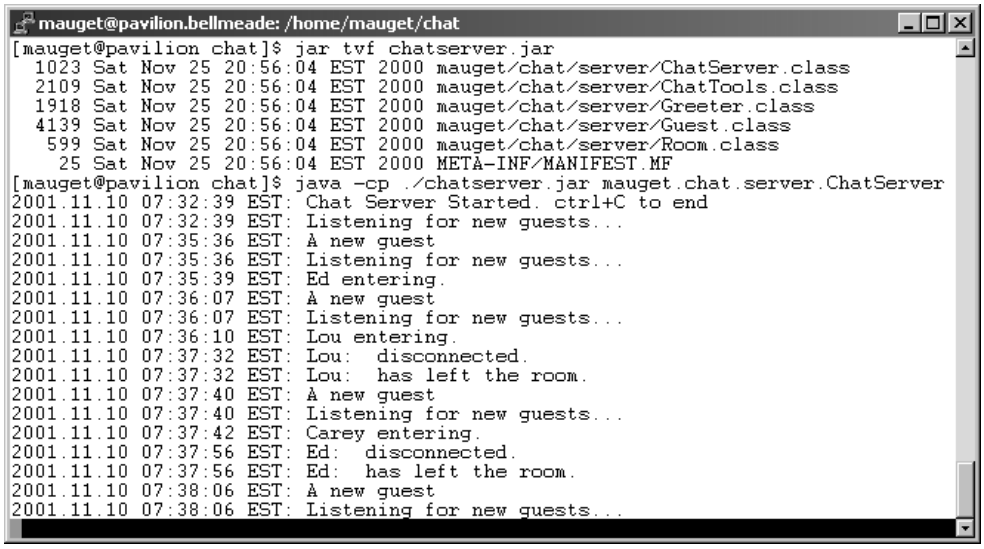

Рис. 6.4. Работающий в среде Linux сервер обмена сообщениями, обращение к которому осуществляется при помощи клиента telnet, запущенного в среде Windows

Лишний символ в конце строки — это самый простой пример несовместимости, которая возникает при запуске приложения Java на разных платформах. Таким образом, мы можем убедиться, что правило «пишем один раз, исполняем на любой платформе» (write once run anywhere) выполняется далеко неидеально. Чтобы исправить данную простую проблему, необходимо удалить из приложения символы возврата каретки. Реализация Java для Windows не использует эти символы, а в Linux они тем более не нужны.

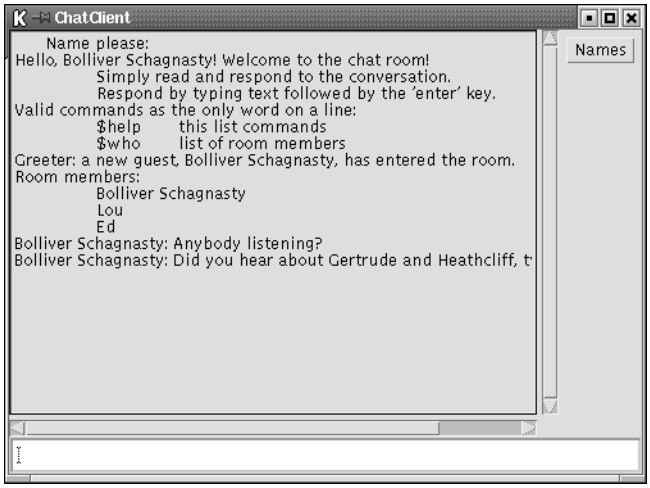

Рис. 6.5. Клиент обмена сообщениями, работающий в Linux

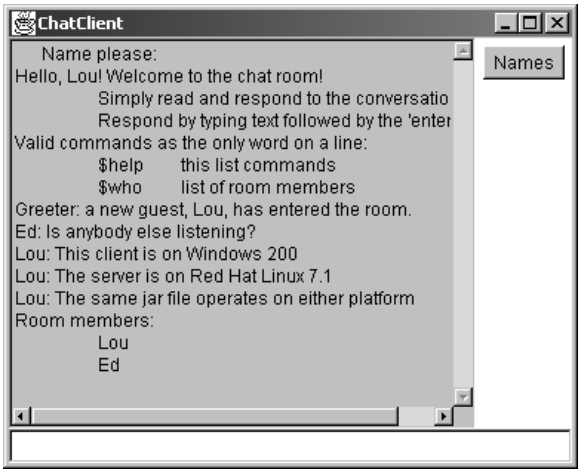

Рис. 6.6. Клиент обмена сообщениями, работающий в Windows

Данный пример демонстрирует, что мантра «пишем один раз, исполняем на любой платформе» (write once run anywhere) вовсе не означает, что мы можем избежать тестирования программы Java на каждой из платформ, на которых планируется использовать наше приложение. Существует лучшее решение: мы могли бы переписать наш сервер в виде комбинации сервлет/JSP-сценарий и запустить его на web-сервере приложений Java. В этом случае тестовой средой становится сервер приложений. Наше приложение будет работать везде, где работает сервер приложений. В качестве клиента может использоваться браузер, который будет всего лишь отображать HTML-код, генерируемый сценарием JSP (то есть сервлетом). Таким образом, клиентом нашего приложения может быть любой компьютер, оснащенный стандартным браузером.

Сетевое программирование

Традиционно сетевые приложения разрабатывались в соответствии с двухзвенной (two-tier) моделью «клиент — сервер», на которой основан, например, только что рассмотренный сервер обмена сообщениями. Служба ожидает поступления запросов через именованный канал (named pipe) или хорошо известный IP-порт. Говорят, что служба «прослушивает» (listen) канал или порт. Когда поступает клиентский запрос, служба генерирует процесс или программный поток, который осуществляет декодирование и обслуживание запроса. Зачастую вместо этого применяется более эффективная методика, подразумевающая, что процесс или программный поток извлекается из набора заранее созданных процессов или потоков, которые уже находятся в памяти и готовы к обслуживанию клиентов. Благодаря тому что процессы или потоки создаются заранее, экономится время, необходимое для создания нового процесса или потока.

Термины «*тонкий клиент*» (thin client), «*толстый клиент*» (fat client) и «*насыщенный клиент*» (rich client) отражают насыщенность клиентского пользовательского интерфейса и то, насколько большая доля обработки данных выполняется на стороне клиента. Разработчик сетевого приложения должен самостоятельно разделить нагрузку и обязанности по обработке данных между клиентом и сервером; кроме того, при разработке двухзвенных сетевых приложений программист вынужден иметь дело с относительно низкоуровневым программированием сокетов или именованных каналов.

В настоящее время существуют высокоуровневые программные интерфейсы, библиотеки и даже развернутые программные инфраструктуры, которые изолируют само по себе приложение от деталей сетевого программирования. Двухзвенные приложения ушли в прошлое, и им на смену пришли трехзвенные (three-tier) приложения. Такие приложения обладают высокой степенью масштабируемости, их проще разрабатывать, внедрять и сопровождать. Новая концепция разработки программ получила название «web-программирование». Вместо того чтобы писать низкоуровневые сокетные приложения, в компании Elsolutions мы сконцентрируем свои усилия на web-программировании. Мы должны максимизировать эффективность взаимодействия между системами Red Hat Linux 7.1 и Windows XP.

Взаимодействие через сеть

В следующих разделах содержится обзор областей обеспечения сетевого взаимодействия и совместимости приложений Java, работающих в системах Windows XP и Red Hat Linux 7.1 компании Elsolutions.

Компоненты

Компоненты — это модули бинарного кода, которые можно использовать повторно. Компоненты можно комбинировать в процессе исполнения программы.

Обращение к компонентам осуществляется через программные интерфейсы. Интерфейсы определяются в процессе разработки компонента, реализация методов компонента скрывается. Описание интерфейса можно получить динамически, в процессе работы приложения. Благодаря этому инструмент разработки программ может связывать несколько компонентов воедино для того, чтобы создать приложение или функцию. Давайте рассмотрим примеры компонентов и варианты их применения в компании Elsolutions.

Компания Microsoft предлагает использовать компоненты COM, которые могут применяться только на платформах Microsoft. Вы можете обращаться к компонентам Microsoft COM вне зависимости от того, на каком языке программирования вы пишете свое приложение. Другими словами, компоненты Microsoft COM не зависят от языка. Для взаимодействия удаленных компонентов COM через сеть используется протокол DCOM. Не существует такого понятия, как компонент DCOM. На самом деле DCOM — это протокол вызова удаленных процедур, предназначенный для обеспечения взаимодействия компонентов COM. Реализация Java компании Microsoft позволяет создавать и использовать компоненты COM, однако компания Microsoft публично объявила о том, что Java не будет входить в состав Windows XP. Термином ActiveX обозначается элемент управления, реализованный в виде компонента COM. Продукт IBM AlphaWorks содержит в своем составе средство Bridge2 Java, которое позволяет обращаться к компонентам ActiveX из программ, написанных на Sun Java, однако только в среде Windows. Существует также средство ActiveX Bridge, которое решает обратную задачу: оно позволяет программисту Visual Basic COM или C++ COM обращаться к компонентам JavaBeans как к элементам управления ActiveX. В Интернете существует пакет DCOM, не основанный на Java, который может работать в среде Linux. Судя по всему, технология COM вряд ли может оказаться полезной при построении сетевой компонентной архитектуры в компании Elsolutions, так как мы планируем использовать в рамках этой архитектуры не только Windows, но и большое количество систем Linux.

В настоящее время индустриальным стандартом инфраструктуры компонентов является технология CORBA. Распределенные, независимые от языка компоненты CORBA взаимодействуют при помощи механизма ORB (Object Request Broker). Используемый в рамках CORBA сетевой протокол IIOP является основным средством взаимодействия приложений Java с произвольными удаленными компонентами через сеть. Применяемая в Linux графическая рабочая среда Gnome использует технологию CORBA подобно тому как Windows использует технологию COM. В обоих случаях CORBA и COM используются для связывания и встраивания дополнительных элементов. Технология CORBA может оказаться полезной для программистов Elsolutions в случае, если потребуется обеспечить взаимодействие приложений Java со службами C/C++ и наоборот.

Ранее мы уже упоминали об используемых в среде Java компонентах EJB (Enterprise JavaBeans). Интересно отметить, что для взаимодействия компонентов EJB через сеть используется механизм RMI (Remote Method Invocation), основанный на протоколе CORBA IIOP. Модуль EJB используется как компонент, однако на самом деле EJB — это несколько классов, добавленных в контейнер EJB. В компании Elsolutions мы планируем использовать по крайней

мере один сервер приложений Java, поэтому имеет смысл использовать при разработке приложений Java компоненты EJB. Недостатком компонентов EJB является то обстоятельство, что для их хранения и использования требуется коммерческий сервер приложений J2EE. По этой причине во многих ситуациях более привлекательной является модель «сервлет — JSP», которая основана на использовании свободно распространяемых серверов приложений, таких как Apache Tomcat. Мы обсудим связанные с этим вопросы далее, в разделах «Сервлеты Java» и «Java Server Pages».

Компоненты JavaBeans впервые появились в версии Java 1.1, в составе которой был реализован интерфейс отражения (reflection API). Чтобы создать компонент JavaBean, необходимо всего лишь написать класс, удовлетворяющий небольшому специальному набору правил. Описание интерфейса компонента JavaBean можно получить при помощи интерфейса отражения. Компоненты JavaBeans используются в качестве связующих элементов в корпоративных приложениях. Они применяются в качестве используемых на стороне клиента оболочек для компонентов EJB, чтобы скрыть относительно сложные процедуры создания экземпляров EJB и доступа к EJB. В приложениях типа «сервлет — JSP» компоненты JavaBeans передаются через сеть в качестве контейнеров для данных. Агрегация компонентов JavaBeans может быть сериализована и десериализована во внешние потоки данных, возможно, даже обладающих форматом XML. Наконец, компоненты JavaBeans могут быть визуально связаны для формирования графических пользовательских интерфейсов Java, выступая в качестве контроллеров и моделей. Компоненты JavaBeans, безусловно, будут играть важную роль при построении интерфейсов и распределенных приложений компании Elsolutions.

Все рассмотренные компоненты могут использоваться в обоих средах — Windows и Linux — однако следует учитывать, что компоненты COM могут функционировать только в среде Windows, а приложения Linux могут обращаться к компонентам COM только через сеть с использованием для этой цели протокола DCOM.

Распределенные приложения

Доступ к удаленным компонентам через сеть — это один из ключей, необходимых для создания распределенных сетевых приложений. Конечно же, распределенное приложение можно создать на базе старой доброй комбинации HTML/HTTP/CGI. Сетевая библиотека Java содержит в себе классы, позволяющие вам кодировать протокол HTTP вручную. Конечно же, это будет работать как в Windows, так и в Linux.

Встроенная в сервер Apache реализация протокола SOAP (Simple Object Access Protocol) написана на Java, что позволяет создавать распределенные web-службы с использованием механизма RPC (Remote Procedure Call), основанного на XML. При этом в качестве транспортного протокола можно использовать HTTP или SMTP. То есть вы можете осуществлять вызовы RPC при помощи электронной почты! Чтобы снизить нагрузку на каналы связи и повысить быстродействие приложения, количество обращений к методам следует минимизи-

ровать — для этого за одну операцию обмена передается как можно больший объем информации. Протокол SOAP агрегирует несколько обращений к методам в единый XML-пакет. Когда в Интернете появятся новые web-службы, основанные на SOAP, этот протокол будет играть важную роль в деятельности компании Elsolutions.

Огромным преимуществом SOAP является то обстоятельство, что вызываемая служба не обязана знать, что обращение к ней осуществляется через сеть. Службу можно реализовать в виде локального класса Java, а затем добавить ее на сервер приложений, разместив описание этой службы в конфигурационном файле XML. Сервлет, выполняющий функции маршрутизатора RPC, преобразует удаленный запрос в обращение к локальному методу и передаст методу полученные через сеть параметры. Результат работы метода будет упакован в формате XML и передан обратно клиенту. Сериализации могут быть подвергнуты сколь угодно сложные параметры. Специальный класс выполняет обработку компонента JavaBean, переданного в качестве параметра. Примитивные и строковые типы обрабатываются без посторонней помощи. Конечно же протокол SOAP не зависит от платформы и используемого языка программирования; благодаря этому обращение через SOAP может быть выполнено между двумя платформами, для взаимодействия которых сложно использовать какие-либо другие технологии. Таким образом, SOAP — это чрезвычайно эффективный механизм обеспечения взаимодействия. Далее приводится пример кода, написанного на Java, который обращается к службе, возвращающей список городов, штатов, кодов Zip, а также телефонных междугородних кодов по первым нескольким буквам имени города. Параметр передается в виде компонента JavaBean. Механизм сериализации компонента JavaBean осуществляет преобразование этого компонента в формат XML и обратно. Хранящийся на сервере XML-дескриптор службы позволяет маршрутизатору XML узнать имя метода и набор параметров, необходимых для обращения к методу. Обратите внимание, что целевая служба указывается при помощи URI: urn:LocationService. Это ключ реестра, используемый маршрутизатором RPC для получения информации о web-службе.

```
/**
 * Получить разнообразную информацию для города, чье имя начинается с cityPrefix
 * Дата создания: (2/13/2001 1:29:14 PM)
 * @return java.util.Vector
 * @param cityPrefix java.lang.String
 */
public Vector citiesLike(String cityPrefix, URL rpcUrl)

    // Формируем запрос
    Call call = new Call();

    // Получаем реестр соответствий и сериализатор bean-компонентов.
    SOAPMappingRegistry smr = new SOAPMappingRegistry();
    BeanSerializer beanSer = new BeanSerializer();

    // Используя сериализатор bean-компонентов, выполняем отображение
    //     компонента Location (местоположение)
    smr.mapTypes(Constants.NS_URI_SOAP_ENC,
                      new Qname("urn:Location", "City"),
```

```
                        com.ibm.mauget.location.Location.class,
                        beanSer,
                        beanSer);
        call.setSOAPMappingRegistry(smr);

        // Устанавливаем целевой объект, метод и стиль кодирования
        call.setTargetObjectURI("urn:LocationService");
        call.setMethodName("citiesLike");
        call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);

        Vector params = new Vector();
        params.addElement(new Parameter("cityPrefix", String.class, cityPrefix, null));
        call.setParams(params);

        Response resp = null;

        // Выполняем обращение к службе
        try {
              resp = call.invoke(/* URL маршрутизатора RPC */ rpcUrl, /* действие URI */
""")ж
        }
        catch (SOAPException e) {
              System.err.println("Caught SOAPException (" +
                    e.getFaultCode() + "): " +
                    e.getMessage());
              return null;
        }

        // Извлекаем полученный в результате обращения массив
        Parameter result = resp.getReturnValue();
        Object[] locs = (Object[])result.getValue();

        // Записываем преобразованный массив в вектор
        Vector values = new Vector(locs.length, 5);
        for (int i = 0; i < locs.length; ++i)
              values.add(locs[i]);
        return values;
  }
```

Обращение клиентов Java к удаленным методам сервера Java используется протокол RMI (Remote Method Invocation). Эта технология проста в настройке и использовании; кроме того, она вполне соответствует философии Java. В качестве службы имен используется реестр RMI, однако вместо JNDI можно использовать другой интерфейс доступа к службе имен. В компании Elsolutions протокол RMI будет использоваться только в качестве среднего звена и только для доступа к внутренним службам, которые реализуют логику среднего звена.

Как уже было отмечено ранее, для взаимодействия между приложениями Java и программами, не написанными на Java, используется протокол CORBA IIOP. Благодаря CORBA IIOP программы, написанные на других языках, могут обращаться к приложениям Java и наоборот. Протокол RMI может использоваться поверх IIOP для обеспечения взаимодействия типа «Java — Java» через механизм CORBA. В компании Elsolutions протокол IIOP может использоваться в случае если компонент среднего звена должен выполнить удаленное обращение к распределенному компоненту C или C++.

Доступ к данным DBMS

Зачастую данные, с которыми работают приложения, приходится хранить достаточно длительное время. В этом случае продолжительность времени хранения данных превышает длительность жизненного цикла программ, которые производят эти данные или работают с ними. Как правило, такие данные хранятся в виде таблиц в реляционных базах данных. Механизм управления этими данными называется системой управления базой данных (СУБД) или, по-английски, DBMS (Data Base Management System). JDBC — это разработанный компанией Sun платформонезависимый интерфейс доступа программ к данным, хранящимся в DBMS. Наименование JDBC часто расшифровывают как Java DataBase Connectivity, однако компания Sun не утверждает, что JDBC является аббревиатурой.

JDBC абстрагирует наиболее важные методы работы с базами данных и группирует их в интерфейс обслуживания базы данных. Поставщик базы данных или сторонний разработчик реализует этот интерфейс в виде драйвера данной конкретной DBMS и, таким образом, становится провайдером службы JDBC. Существует четыре типа драйверов JDBC:

- Драйверы Microsoft ODBC API, которые используются также через драйвер моста JDBC-ODBC первого типа (*Type 1 JDBC-ODBC Bridge Driver*). В данном случае инструкции передаются от JDBC к ODBC, что несколько замедляет их обработку, однако благодаря использованию данного моста вы получаете возможность работать с системами DBMS, для которых нет прямого драйвера JDBC. Еще одним недостатком моста JDBC-ODBC является необходимость использования программного обеспечения, не написанного на Java (часть, имеющая отношение к драйверу ODBC).
- Драйвер естественного API второго типа (*Type 2 Native-API Driver*) — тонкий уровень кода Java, который инкапсулирует родную библиотеку DBMS. Например, IBM DB2 CLI можно инкапсулировать при помощи оболочки JNI (Java Native Interface); в результате получится драйвер второго типа. Такой драйвер может работать быстрее, чем драйвер, целиком и полностью написанный на Java, однако следует учитывать, что драйверы второго типа привязаны к определенной платформе.
- Драйвер сетевого протокола третьего типа (*Type 3 Net-protocol Driver*) — целиком и полностью пишется на Java. Для обращения к базе данных он взаимодействует с произвольным уровнем программного обеспечения среднего звена.
- Драйвер, полностью написанный на Java четвертого типа (*Type 4 All-Java Driver*), напрямую взаимодействует с DBMS и понимает ее прямые сетевые протоколы.

Драйверы третьего и четвертого типов могут использоваться апплетами, так как они могут загружаться и исполняться в рабочей среде web-браузера (подразумевается, что диспетчер безопасности Java разрешает сетевые соединения с DBMS).

Как показано в следующем примере, перед выполнением каких-либо операций JDBC приложение JDBC обязано загрузить класс драйвера JDBC для базы данных, с которой оно работает. Стандарт JDBC играет важную роль для обеспечения совместной работы с данными в средах Windows XP и Red Hat Linux 7.1. Клиент доступа к данным, написанный на Java и использующий JDBC, может работать на любой платформе — в особенности если он использует драйверы JDBC третьего и четвертого типов.

```
try {
    // Регистрируем драйвер MySql
    Class.forName("org.gjt.mm.mysql.Driver");
}
    catch (ClassNotFoundException ex) {
    System.err.println("Mysql JDBC class didn't load");
    ex.printStackTrace();
    return;
}
```

Прежде чем можно будет выполнить какую-либо операцию в отношении базы данных, необходимо установить с ней соединение. Соединение инкапсулируется в экземпляре объекта java.sql.Connection, который можно получить от диспетчера драйвера JDBC. Адрес URL, именующий JDBC как сетевой протокол, определяет местоположение DBMS, а также указывает имя базы данных, используемое для подключения к базе данных. Операция DBMS запрашивается с использованием методов экземпляра класса java.sql.Statement. Экземпляр соединения — это фабрика классов, которая используется для создания объектов этого класса. Типичная последовательность действий, выполняемых для создания объектов java.sql.Connection и java.sql.Statement, выглядит следующим образом:

```
Import java.sql.*;
...
Connection con;
Statement stmt;
...
try {
    con = DriverManager.getConnection("jdbc:mysql:Zipcodes");
    stmt = con.createStatement();

}
catch (SQLException ex) {
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
    ex.printStackTrace();
    return;
}
```

Несмотря на то, что интерфейс JDBC абстрагирует операции с базами данных, на самом деле он удален от них незначительно. Чтобы работать с JDBC, вам потребуется знание SQL. Интерфейс JDBC можно использовать как для команд определения схемы базы данных, так и для команд манипулирования данными. В следующем коде мы используем выражение определения схемы SQL для того чтобы создать таблицу. Так как мы осуществляем запись в базу данных, для передачи выражения SQL системе DBMS используется метод

java.sql.Statement.executeUpdate(). Этот метод используется также для того чтобы заполнить таблицу данными или для того чтобы изменить данные в строке таблицы.

```
try {
    // Создать в базе данных, с которой установлено соединение,
    //    таблицу location для хранения
    //    информации о географическом местоположении
    stmt.executeUpdate("CREATE TABLE location " +
"(id int not null auto_increment, primary key (id)," +
"city varchar(40) not null," +
"state char(2) not null," +
"zip char(10) not null," +
"area char(3))"
    );

    // Вставляем в таблицу location новую строку
    stmt.executeUpdate("INSERT location values (" +
    "0, 'Spokane','WA','99207','509')"
    );
}
catch (SQLException ex) {
    System.out.println("SQLException: " + ex.getMessage());
}
```

Большая часть работы с DBMS заключается в исполнении запросов к базе данных. Запрос извлекает информацию из колонок таблицы, переходя от строки к строке. Запрашивающая программа осуществляет последовательный перебор строк таблицы, используя курсор базы данных. JDBC инкапсулирует результирующую таблицу в экземпляре класса java.sql.ResultSet. Этот объект возвращается методом java.sql.Statement.executeQuery().

```
try {
    // Запрашиваем строки, содержащие zip-код 99207 из таблицы location
    ResultSet rs =
      stmt.executeUpdate("SELECT city, state FROM location " +
   "WHERE zip = '99207'"
     );

     while(rs.next())
         System.out.println(rs.getString("city") +
              ", " +
              rs.getString("state"));

      // Освобождаем ресурсы
      rs.close();
}
catch (SQLException ex) {
    System.out.println("SQLException: " + ex.getMessage());
}
```

Чтобы освободить ресурсы, все объекты java.sql.Connection и java.sql.Statement необходимо закрыть. Соединение необходимо закрыть в последнюю очередь:

```
try {
    stmt.close();
    con.close();
}
```

```
catch (SQLException ex) {
    System.out.println("SQLException: " + ex.getMessage());
}
```

Интерфейс JDBC является достаточно низкоуровневым, так как представляет собой тонкую абстракцию DBMS API. Существуют программные системы, которые позволяют абстрагировать объекты долговременного хранения в виде внутренних объектов Java. Система Strata Framework™, разработанная компанией CrossLogic Corporation, предлагает два основанных на JDBC метода отображения данных долговременного хранения в объекты данных Java. Спецификация Sun Enterprise Java Beans (EJB) содержит в своем составе компонент, время жизни которого может превышать время жизни экземпляров работающих с ним приложений. В качестве места долговременного хранения компонента используется DBMS, доступ к которой осуществляется с использованием JDBC. Благодаря такому подходу бизнес-логику приложения удается изолировать от деталей, связанных с реализацией механизма хранения данных. В любом случае независимый от платформы доступ к DBMS реализуется с использованием JDBC.

Очереди сообщений

При формировании распределенных приложений в качестве объединяющей среды часто используется технология обмена сообщениями (Message-Oriented Middleware, MOM). Очередь сообщений — это место, куда можно добавить сообщение и из которого можно извлечь сообщение. Фактически данный механизм выполняет функции электронной почты для разных прикладных программ. Механизм обмена сообщениями настолько универсален, что с его помощью можно преодолеть любые сложности, связанные с обеспечением взаимодействия самых разнообразных сетевых платформ и приложений. Например, агент по продажам, использующий клиента для работы с очередью сообщений, работающего на портативном компьютере, может вносить заказы в базу данных DB2, работающую на мейнфрейме IBM. Для целей нашей книги можно представить себе другой сценарий: пользователь портативного компьютера Windows XP вносит заказы в базу данных, работающую на сервере Red Hat Linux 7.1.

Функциональные элементы подобного решения показаны на рис. 6.7. Агент по продажам, работающий с портативным компьютером, может создавать новые заказы на поставку товара в течение рабочего дня, путешествуя по офисам заказчиков. Каталог товаров содержится в локальной базе данных DB2 на портативном компьютере Windows XP. Когда агент создает новый заказ (заполняя для этой цели электронную форму), этот заказ попадает в локальную очередь сообщений. Используемое для этой цели клиентское приложение может быть приложением Java, которое использует JMS (Java Message Service) для доступа к локальной очереди сообщений и JDBC доступа к локальной базе данных.

В конце рабочего дня агент по продажам подключается к серверу Linux (в нашем случае для этого используется модем). Очередь управляется специальной программной системой, такой как IBM webSphere Messaging (по-другому эту систему называют MQSeries). Такое программное обеспечение называют

программным обеспечением среднего звена (middleware). Сервер выполняет операции message get до тех пор, пока очередь, хранящаяся на портативном компьютере Windows XP, не станет пустой. Серверная сторона нашего распределенного приложения обрабатывает каждый из заказов. При этом для каждого заказа обратно клиенту отсылается подтверждающее сообщение — для этого опять же используется очередь сообщений. Если модемное соединение с клиентом разрывается, подтверждающее сообщение продолжает храниться в очереди сообщений сервера до тех пор, пока оно не будет передано клиенту во время следующих соединений.

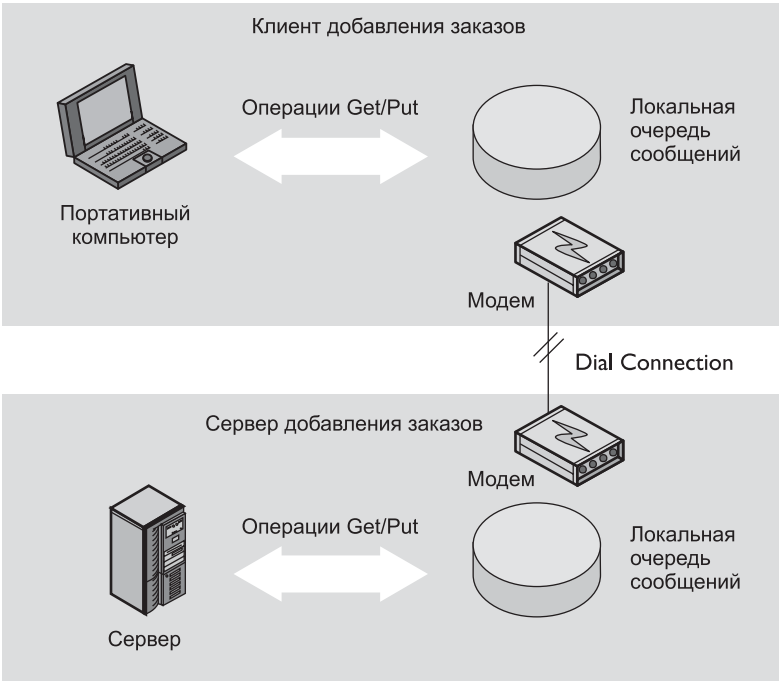

Рис. 6.7. Приложение добавления заказов в базу данных

Какими преимуществами наделяет нас система очередей сообщений? Благодаря асинхронной природе соединения достигается высокая степень гибкости при выполнении клиент-серверных операций в случае если соединение непостоянно. Программное обеспечение среднего звена позволяет нам асинхронно соединять разнородные приложения простым способом, без применения сложного протокола. Конечные точки нашего распределенного приложения знают только о том, как открыть очередь и как выполнить операции get и put, определенные в реализации языка MOM (такой как, например, JMS). Конечно же, для того чтобы установить соединение и выполнить операции манипулирования очередью, потребуются разнообразные настройки и параметры, однако пользовательский код работает только с протоколом get/put.

Еще одной стороной обработки сообщений является гарантированная доставка сообщений. Представьте себе того же самого агента, формирующего заказы в то время, как его портативный компьютер не имеет связи с сервером. Когда компьютер торгового агента подключается к серверу, поставленные в очередь заказы принимаются и обрабатываются сервером. Именно в это время агент узнает о любых дополнительных условиях (например, отсутствие товара на складе).

Каким образом Java вписывается в эту схему? Неудивительно, что все наиболее крупные поставщики решений в этой области обеспечивают взаимодействие своих решений с технологией Java. Благодаря этому эти решения могут использоваться как в Windows, так и в Linux. Основным игроком на рынке программных решений в области обмена сообщениями является система IBM MQSeries, а Java является центром программной стратегии IBM. Но может ли Java работать с продуктами MOM других поставщиков?

Для этого существует JMS (Java Messaging Service). JMS компании Sun — это набор интерфейсов Java, описывающих архитектуру провайдера услуг по обработке сообщений. Каждый крупный поставщик решения в области обмена сообщениями создает свою собственную реализацию интерфейса JMS, поэтому, загрузив JMS компании Sun, вы будете разочарованы, так как эта реализация ориентирована только на продукты Sun. Таким образом, чтобы работать с JMS, вы должны загрузить продукт конкретного поставщика. JMS обеспечивает взаимодействие между поставщиками решений MOM, в которых реализован этот интерфейс. Конечно же, этот интерфейс можно использовать как в Windows, так и в Linux.

Служба имен и служба каталога

Мы уже упоминали об интерфейсе JNDI (Java Naming and Directory Interface), когда речь шла о компонентах EJB. Было отмечено, что компоненты EJB регистрируются и обнаруживаются при помощи провайдера JNDI. Как и JMS, интерфейс JNDI является интерфейсом провайдера услуг. Если вы хотите работать с JNDI, вы должны указать провайдера, доступ к которому будет осуществляться через этот интерфейс. В составе J2SE присутствуют несколько стандартных провайдеров: LDAP, COS (служба имен CORBA) и реестр RMI. Отдельно поставляются провайдеры LDAP Booster Pack, NIS, а также провайдеры доступа к файловым системам. В то время, когда писалась эта книга, была доступна предварительная версия провайдера DNS. Благодаря такому провайдеру приложения Java смогут использовать интерфейс JNDI для доступа к службе доменных имен DNS (Domain Name Service). Также существует предварительная версия провайдера JNDI для доступа к DSML (Directory Services Markup Language).

Благодаря JNDI приложения Java, работающие как в Windows, так и в Linux, получают возможность доступа к многочисленным службам каталогов и службам имен. Дополнительно об этом будет рассказано в главе 15.

XML

Язык XML (Extended Markup Language) — это язык разметки, предназначенный для создания производных языков разметки, которые в свою очередь пред-

назначены для описания сколь угодно сложных данных. XML используется для описания данных. Существует большое количество литературы, посвященной этой новой стремительно развивающейся технологии. Язык XML популярен благодаря тому, что он позволяет передавать данные между совершенно разными процессами и приложениями. Как и Java, технология XML основана на общераспространенных стандартах, именно поэтому она получает все большее распространение. В действительности Java и XML в совокупности являются превосходной комбинацией. Компания IBM разработала множество свежих, доступных для широкой общественности технологий, основанных на комбинации XML плюс Java. Эти решения можно обнаружить либо на принадлежащем компании IBM web-узле AlphaWorks, либо в других источниках, например в организации Apache. Web-узел AlphaWorks располагается по адресу http://alphaworks.ibm.com/.

Грамматический разбор XML-кода можно выполнить несколькими разными способами. Во-первых, для работы с XML-данными можно использовать механизм DOM (Document Object Model). Механизм DOM читает код XML и формирует в оперативной памяти соответствующую структуру данных в виде иерархии записей. Эта структура хранится в памяти резидентно в течение всего времени работы с документом XML. Программный интерфейс DOM используется для перемещения по иерархии и модификации структуры узлов и значений атрибутов. Используя интерфейс DOM, вы можете также программно сформировать свою собственную иерархию, а затем при помощи специальной функции экспортировать эту иерархию в текстовый поток XML.

Второй метод грамматического разбора XML-документов основан на использовании технологии SAX (Simple Access for XML). Механизм SAX не хранит в памяти логическое представление данных XML — вместо этого он обрабатывает код XML как поток данных. SAX последовательно перебирает записи XML и, в соответствии с обнаруженными элементами, генерирует исключения. Обработчики исключений реализуются в рамках клиентского приложения, которое выполняет обработку XML-кода. SAX отлично подходит для обработки больших объемов структурированных данных, например, при загрузке крупной базы данных в приложениях, основанных на обработке событий (таких, как графические пользовательские интерфейсы). Механизм SAX можно использовать для реализации модели DOM. Существует по крайней мере одно такое решение — продукт Apache Xerces. Чтобы обработать XML-код с использованием SAX, программист пишет код обработчиков событий, перекрывая тем самым тривиальные обработчики, используемые в рамках SAX по умолчанию. Разнообразные события генерируются в начале документа, в конце документа, в начале и в конце каждого элемента XML, для различных атрибутов, а также для анализа и извлечения символьных данных.

Структура и конкретный синтаксис документа XML обозначаются термином «схема документа XML». Для описания структуры документа XML может использоваться язык DTD (Data Type Declaration). Говоря иначе, язык DTD используется для описания грамматики или схемы документа XML. DTD — это еще один отдельный язык разметки, основанный на языке SGML (Standardized General Markup Language), на основе которого был в свое время разработан и XML.

Недостатком DTD является то, что это еще один отдельный язык. С DTD связана и еще одна проблема: его возможности по описанию структуры XML ограниченны.

Почему нельзя использовать для описания структуры документа XML сам язык XML? И действительно, существует такой XML-язык, который называется XML Schema. Для преобразования DTD в XML Schema можно использовать редактор IBM Xeena.

Некоторые из механизмов грамматического разбора, основанных на DOM или SAX, являются механизмами, проверяющими корректность XML-кода. Это означает, что они следят за соответствием XML-данных заявленному описанию структуры, определенному при помощи DTD или XML Schema. Если документ не соответствует описанию, генерируется исключение. Другие механизмы грамматического разбора не выполняют проверку корректности. При этом исключение генерируется только в случае если синтаксис XML-документа не удовлетворяет правилам XML.

Основанный на Java механизм грамматического разбора IBM AlphaWorks XML4J положен в основу программного средства Apache Xerces. На самом деле в состав этого продукта входят четыре отдельных средства грамматического разбора XML. В составе XML4J реализованы оба интерфейса — DOM и SAX, кроме того, каждый из них можно настроить на проверку корректности как с использованием DTD, так и с использованием XML Schema. Описание DTD или XML Schema может быть как встроенным в документ XML, так и прилагаться в виде отдельного документа.

Основанный на Java редактор XML под названием IBM AlphaWorks Xeena использует грамматический разбор с проверкой корректности для того чтобы обеспечить создание и редактирование XML-документов, структура которых соответствует описанию DTD или XML Schema. Редактор Xeena можно использовать для преобразования DTD в XML Schema.

Основное предназначение XML — это обеспечение обмена данными между приложениями. Как правило, два произвольных приложения используют данные разной структуры, представленные в разных форматах. Основанная на XML технология под названием XSL (Extensible Stylesheet Language) использует XML для описания соответствий XSL, при помощи которых схема одного документа преобразуется в схему другого документа. Трансформация XSL (XSL Transform, XSLT) основана на использовании еще одного элемента этой технологии — Xpath — и позволяет обращаться к любому узлу структуры данных XML без использования программирования. Таким образом, преобразование XSL применяется к входному XML-документу, и в результате генерируется выходной трансформированный XML-документ. XSLT часто используется для создания страниц HTML на основе данных, описанных с использованием XML. Хорошим инструментом, выполняющим подобные преобразования, является основанное на Java программное средство Xalan от Apache.

HTML используется для разметки визуального документа, однако HTML понятия не имеет о том, какие данные отображаются. С другой стороны, XML используется для описания данных, однако XML не отвечает за визуальное представление этих данных. Визуальное представление данных возлагается на кон-

кретное приложение. Два разных приложения могут отобразить одни и те же данные по-разному.

До последнего времени HTML был в некотором роде ущербным языком. Браузер пытается отобразить HTML-документ даже в том случае, если в нем содержится грамматическая ошибка. В HTML многое декларируется неявно: например, кавычки, ограничивающие значение атрибута, могут быть опущены. С другой стороны, XML подразумевает использование жестко заданного синтаксиса. Если HTML-документ переписать так, чтобы он удовлетворял всем требованиям XML, этот документ по-прежнему останется HTML-документом. В этом случае он будет называться XHTML. В результате трансформации XSLT часто получается документ XHTML или фрагмент кода XHTML, используемый для визуального представления исходных данных.

Например, следующий документ XML описывает два элемента в гипотетическом каталоге:

```
<?xml version="1.0"?>
<ITEMS>
   <PRODUCT>
      <BRAND>KrazyKomputerz</BRAND>
      <CLASS>Printer<TYPE>stone cutter</TYPE></CLASS>
      <NAME>SlamBang2x2</NAME>
      <FEATURES>
         <SPEED Units="ppm">0.1</SPEED>
         <QUALITY Units="dpi">2</QUALITY>
      </FEATURES>
      <PRICE Units="Scheckels">
         <RETAIL>200</RETAIL>
         <WHOLESALE>110</WHOLESALE>
      </PRICE>
      <WEIGHT Units="Dynslotkys">10</WEIGHT>
   </PRODUCT>
   <PRODUCT>
      <BRAND>KopuSub</BRAND>
      <CLASS>Removable Storage<TYPE>Cheapo</TYPE></CLASS>
      <NAME>ZAP Drive</NAME>
      <FEATURES>
         <SPEED Units="rpfn">9</SPEED>
         <QUALITY Units="nybbles">314</QUALITY>
      </FEATURES>
      <PRICE Units="Scheckels">
         <RETAIL>5</RETAIL>
         <WHOLESALE>2</WHOLESALE>
      </PRICE>
      <WEIGHT Units="Stone">10</WEIGHT>
   </PRODUCT>
```

Язык XML обладает поддержкой *пространств имен* (namespaces), благодаря чему удается избежать конфликтов имен. Префикс xmlns используется для замены или указания полного префикса имени. Как правило, для спецификации пространств имен используются адреса URL, однако в этом случае они не ссылаются на некоторый ресурс web, а всего лишь используются для обеспечения уникальности имен в XML-документе. В следующем примере демонстрируется трансформация XSL с применением xmlns. Пространство имен по умолчанию, то есть выражение, за которым не следует символ двоеточия, используется для

указания в результирующем документе обычного пространства имен XHTML. Пространство имен xmlns:xsl указывает элементы XSL в документе. Благодаря этому мы можем смешивать имена, не опасаясь конфликтов. Давайте преобразуем приведенный ранее документ XML в документ HTML. Для этого применим к нему следующую трансформацию:

```
<?xml version="1.0"?>
<html xsl:version="1.0"
    xmlns="http:/www/w3.org/TR/xhtml1/strict"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <head><title>Inventory</title></head>
    <body bgcolor="darkblue" fgcolor="white" text="white">
    <h1>Profound Bargains</h1>
       <xsl:for-each select="ITEMS/PRODUCT">
       <xsl:sort select="BRAND" />
       <table border="1" cellspacing = "0" cellpadding="2">
         <tbody>
               <tr><th align="right">Manufacturer:</th>
           <td width="200"><xsl:value-of select="BRAND" /></td></tr>
           <tr><th align="right">Class:</th>
           <td><xsl:value-of select="CLASS/text()" /></td></tr>
           <tr><th align="right">Type:</th>
           <td><xsl:value-of select="CLASS/TYPE" /></td></tr>
           <tr><th align="right">Name:</th>
           <td><xsl:value-of select="NAME" /></td></tr>
           <tr><th align="right">Speed:</th>
           <td><xsl:value-of select="FEATURES/SPEED" />
                 <![CDATA[ - ]]>
                 <xls:value-of select="FEATURES/SPEED/@Units" />
                 </td></tr>
           <tr><th align="right">Quality:</th>
           <td><xsl:value-of select="FEATURES/QUALITY" />
                 -
                 <xls:value-of select="FEATURES/QUALITY/@Units" />
                 </td></tr>
           <tr><th align="right">Retail:</th>
           <td><xsl:value-of select="PRICE/RETAIL" />
                 -
                 <xls:value-of select="PRICE/@Units" />
                 </td></tr>
           <tr><th align="right">Wholesale:</th>
           <td><xsl:value-of select="PRICE/WHOLESALE" />
                 -
                 <xls:value-of select="PRICE/@Units" />
                 </td></tr>
           <tr><th align="right">Weight:</th>
           <td><xsl:value-of select="WEIGHT" />
                 -
                 <xls:value-of select="WEIGHT/@Units" />
                 </td></tr>
         </tbody>
       </table>
       <br/>
       </xsl:for-each>
       <hr/>
    </body>
</html>
```

Данная трансформация просматривает документ в поисках записей о продуктах <PRODUCT>, расположенных в разделе <ITEMS>. Обратите внимание на цикл for-each и спецификацию sort. На самом деле XSL — это фактически язык программирования, определенный в пространстве имен XML. Это выражение Xpath. Другие выражения XPath используются для обнаружения элементов, которые необходимо внести в ячейки HTML-таблицы. Трансформация XSL применяется в отношении документа XML при помощи Java-программы Xalan от Apache. Для этого необходимо выполнить следующую команду:

```
java org.apache.xalan.xslt.Process -IN product.xml -XSL product.xsl -OUT product.html
```

Полученный в результате этого HTML-документ частично показан на рис. 6.8. Обратите внимание, что ни один из трех документов и даже программное средство Apcahe Xalan никак не привязаны к какой-либо конкретной платформе. Трансформация выполняется с одинаковым успехом как в Windows, так и в Linux. Ввод, трансформация и отражение XML-документов не являются платформо-зависимыми. Это важный пример широких возможностей взаимодействия, которыми наделяют вас Java и XML.

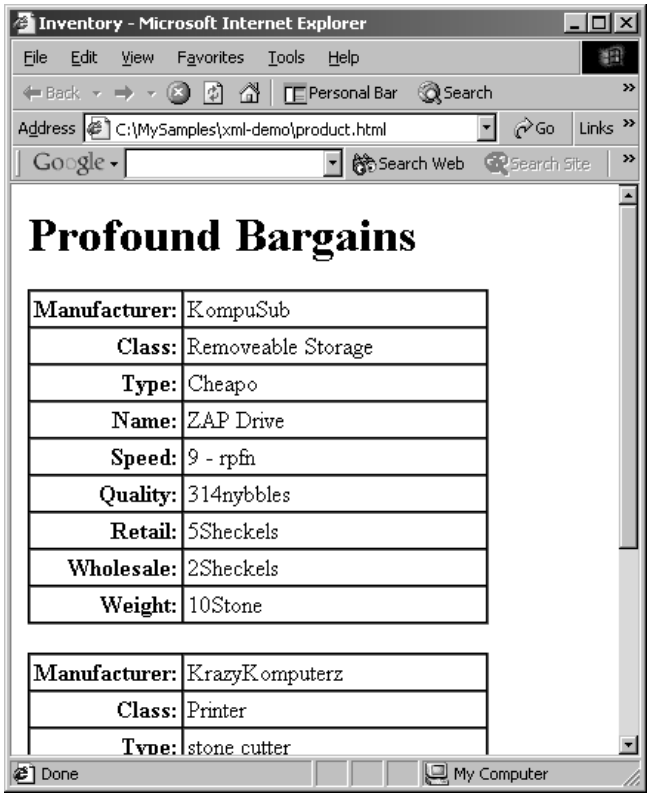

Рис. 6.8. Документ XML, трансформированный в страницу HTML

Преобразование кода XML в визуальные документы HTML — это традиционная область применения XML в web-программировании. В качестве языка, используемого для выполнения операций XML, зачастую используется Java, так как этот язык не зависит от серверной или клиентской платформы. Это означает, что пакет XML, основанный на Java, может использоваться как в среде Windows, так и в среде Linux. XML в комбинации с Java будет играть важную роль при построении распределенных приложений компании Elsolutions для Интернета и интранета.

Интернет, интранет

Java широко используется в программировании Интернета и Web. С использованием Java создаются серверные приложения, а также апплеты, которые загружаются на компьютер клиента и работают в web-браузере. Для построения web-приложений, основанных на Java, используются сервлеты Java, сценарии JSP и компоненты EJB. В дальнейшем мы будем рассматривать эти технологии, а также их значение для компании Elsolutions. Таким образом, сетевое программирование в компании Elsolutions сводится к web-программированию.

Web-программирование

Итак, мы пришли к выводу, что большая часть программирования в компании Elsolutions является web-программированием. Web-программирование — это не так просто, как может показаться на первый взгляд. Если вы просто будете использовать самые свежие технологии в этой области, не обладая при этом нужными навыками, вы можете только лишь усложнить себе жизнь. Основная идея состоит в том, что двухзвенная модель «клиент — сервер», распространенная в 90-х годах прошлого века, ушла в прошлое, и ей на смену пришла расширенная — трехзвенная или даже многозвенная — модель нового тысячелетия. В рамках этой модели клиент выполняет функции презентационного звена, обмен данными с ним осуществляется через сеть. Звено данных или, по-другому, звено долговременного хранения информации — это хранилище, расположенное на противоположном конце цепочки (его часто обозначают английским термином *backend*). Собственно логика приложения размещается в среднем звене. Приложение является посредником между клиентами и службой хранения данных. Это среднее звено поддерживает обслуживание множества клиентов и обеспечивает для них параллельную работу с данными. Однако описанная здесь относительно простая схема на практике может быть более запутанной: в действительности части приложения могут располагаться в любом из звеньев. Например, звено данных может включать в себя сложные процедуры сохранения и извлечения данных; кроме того, клиент может представлять собой комбинацию прикладной логики и сценариев, работающих на стороне клиента, апплетов Java или даже элементов управления ActiveX, работающих в среде Internet Explorer.

Сценарии, работающие на стороне клиента

Часть web-приложения может быть реализована в виде сценария, являющегося частью web-страницы, обрабатываемой браузером. Компания Microsoft изначально называла такие сценарии динамическим HTML (Dynamic HTML, DHTML), однако сейчас эту технологию можно с тем же успехом назвать сценариями DOM, так как фактически сценарий манипулирует кодом HTML или XML при помощи интерфейса DOM, встроенного в браузер. В различных браузерах интерфейс DOM может быть реализован по-разному, поэтому при разработке подобных сценариев могут возникнуть проблемы, связанные с совместимостью. Разработчики web-приложений для Интернета вынуждены прилагать массу усилий для того, чтобы обеспечить совместимость с разнообразными типами браузеров, используемых в Интернете. Если речь идет о разработке web-приложения для конкретного предприятия, то задача упрощается, так как в каждой компании, как правило, используется один стандартный web-браузер.

Для каких целей можно использовать сценарии, работающие на стороне клиента? Сценарий DOM может выполнять простую проверку правильности данных, вводимых пользователем в поля HTML-формы, или генерировать специальные видеоэффекты пользовательского интерфейса. Проверка правильности вводимых пользователем данных, выполняемая на стороне клиента, снижает нагрузку на каналы связи, так как в случае если при вводе этих данных пользователь допустит ошибку, эта ошибка будет обнаружена прямо в браузере, то есть ошибочно введенные данные не будут передаваться в сеть. Передача данных в сеть будет выполнена только в случае, если пользователь ввел в поля формы HTML корректную информацию. Если HTML-код динамически генерируется кодом сервлета Java, код сервлета может встроить в HTML специальный сценарий, осуществляющий необходимую проверку. Таким образом можно генерировать специализированные элементы управления для web-интерфейса. Например, предположим, что сервлет осуществляет некоторую обработку данных на основе введенного пользователем Zip-кода (Zip-код в США — это нечто вроде почтового индекса (*примеч. переводчика*)). Для получения Zip-кода сервлет генерирует HTML-код web-страницы, на которой располагается текстовое поле для ввода этого Zip-кода. В составе этого кода помимо описания текстового поля содержится сценарий, выполняющий проверку правильности пользовательского ввода. Если пользователь вводит что-либо, не удовлетворяющее формату Zip-кода, сценарий отображает на экране сообщение об ошибке и просит пользователя повторить ввод. Если в качестве Zip-кода указано число, формат которого соответствует формату Zip-кода, это число передается через сеть сервлету, который выполняет дальнейшую обработку.

Не следует использовать на стороне клиента слишком большое количество сценариев, так как в результате получается слишком громоздкое приложение. Если вы используете клиентские сценарии для чего-либо еще помимо проверки правильности ввода и спецэффектов GUI, это значит, что, скорее всего, вы делите логику приложения между клиентом и сервером. Подобное разделение может усложнить вам жизнь.

Сценарии, работающие на стороне клиента, как правило, пишутся на JavaScript или ECMA Script, так как эти языки поддерживаются большинством браузеров как в Windows XP, так и в Red Hat Linux 7.1. В компании Elsolutions планируется использовать сценарии, работающие на стороне клиента, для проверки вводимых данных и для спецэффектов GUI, а основную логику web-приложений планируется целиком размещать на сервере приложений.

Апплеты Java

Апплеты Java обсуждаются в главе 10. Как и сценарий, работающий на стороне клиента, апплет Java — это программа, которая исполняется на клиентском компьютере. Апплет может использоваться для генерации спецэффектов; кроме того, он может выполнять функции толстого клиента (fat client) по отношению к приложению, работающему на сервере. Благодаря использованию апплетов вы можете избежать проблем, связанных с развертыванием web-приложений в сети. Такие проблемы были характерны для традиционных двухзвенных клиент-серверных приложений. При использовании в качестве клиентской части апплета Java на клиентском компьютере не требуется выполнять какую-либо специальную процедуру установки: апплет автоматически загружается с сервера каждый раз, когда в этом нуждается клиентский браузер. Это обстоятельство является одновременно и благословением, и проклятьем. С одной стороны, такой подход значительно облегчает установку, обновление и контроль различных версий клиентской части приложения. С другой стороны, каждый раз, когда клиент нуждается в апплете, его приходится передавать через сеть, при этом размер апплета может быть достаточно большим.

Еще одна проблема, связанная с апплетами Java, состоит в том, что в разных браузерах могут использоваться разные версии JVM. В течение длительного времени в мире не существовало ни одного браузера, поддерживающего Java 1.1. Позже появились браузеры, совместимые с JVM 1.1, однако значительная часть браузеров по-прежнему позволяет использовать только Java 1.0. Разработчик апплетов Java не может рассчитывать на новые возможности свежих версий Java, так как значительный сегмент клиентских браузеров не поддерживает их. К счастью, проблему отчасти решает изобретение встраиваемого в браузер модуля JVM (JVM plug-in). Благодаря этому сервер может контролировать как код, исполняемый на стороне клиента, так и версию JVM, которая используется для исполнения этого кода.

Апплет Java исполняется под контролем строгих механизмов безопасности, обеспечивающих защиту клиентской машины от злоумышленных действий и непреднамеренных ошибок. Это означает, что возможности апплета в значительной степени ограниченны: ему не разрешается делать многих действий, например, обращаться к локальным файлам клиентского компьютера. Однако в случае необходимости уровень защиты можно понизить: для этого используется диспетчер безопасности Java.

В наши дни многие предприятия отказываются от использования апплетов Java в пользу трехзвенных приложений. Разработчики компании Elsolutions также решили отказаться от широкого применения апплетов Java, однако аппле-

ты Java решено использовать в некоторых редких случаях — например, для создания визуальных и аудиоэффектов для интернет-приложений.

Трехзвенные приложения

На смену клиент-серверному программированию пришла трехзвенная или многозвенная архитектура. В рамках этой технологии приложение, содержащее в себе бизнес-логику, играет роль посредника между клиентом и DBMS. Можно сказать, что приложение, содержащее в себе бизнес-логику, является клиентом по отношению к DBMS и сервером по отношению к клиентскому компьютеру. Клиентский web-браузер выполняет функции пользовательского интерфейса. Комбинация этих трех составляющих называется трехзвенным приложением. Три звена — это клиент, сервер и служба хранения данных. Многозвенным называется приложение, в котором одно из описанных звеньев распределено между несколькими компьютерами. Многозвенным также можно назвать приложение, в котором используется механизм распределения нагрузки (load balancing). Термином web-ферма (web farm) называют большое количество компьютеров, которые формируют собой единое web-приложение (или несколько взаимосвязанных приложений).

Существует множество вариаций прикладных инфраструктур, методик использования сценариев, моделей развертывания web-приложений, а также инструментов, используемых для создания web-приложений. Web-приложения являются областью, в которой взаимодействие и взаимосовместимость играют большую роль. Для создания web-фермы или распределенного приложения могут использоваться самые разные комбинации компьютеров Windows и Linux.

Распределенными могут быть самые разные приложения, например, организационные приложения дискуссионных порталов или мощные масштабируемые корпоративные решения на базе Интернета и интранета, обращающиеся к крупным базам данных. Используемые для этой цели интерфейс API и web-технология определяются функцией приложения, масштабом и экспертными оценками. Трехзвенные приложения будут играть значительную роль в компании Elsolutions. Эта архитектура будет использоваться как для приложений интранет, так и для решений в области Интернета. Рассмотрим программные технологии, используемые для создания среднего звена трехзвенных web-приложений.

Программные технологии среднего звена

Мы уже говорили, что среднее звено можно рассматривать как посредника между клиентом и данными. Можно представить также, что браузер — это GUI (пользовательский интерфейс), а среднее звено — это основное логическое ядро (engine) web-приложения. Вспомним классическую модель «модель — представление — контроллер» (Model — View — Controller, MVC). Функции представления берет на себя браузер, а среднее звено делится на две части: модель и контроллер. Контроллер осуществляет взаимодействие с браузером, а модель выполняет

функцию сценария обработки бизнес-правил. В крупных предприятиях бизнес-правила могут быть реализованы в виде библиотек, допускающих повторное использование. Такие библиотеки могут состоять из классов или компонентов. Напротив, в несложных приложениях модель и представление могут быть объединены в единый набор объектов или в один кусок кода.

Выбор программной технологии или общего дизайна будущего web-приложения сопряжен с затратами. Зачастую некоторый дизайн может показаться неуклюжим при создании простой или демонстрационной программы, однако этот же дизайн может оказаться наиболее эффективным в случае, если вы захотите использовать его для создания достаточно крупного трехзвенного приложения. Далее мы будем обозначать термином «web-программирование» процесс программирования трехзвенных приложений.

Мы предполагаем, что с клиентской стороной все более-менее понятно. На стороне клиента для отображения выводимой информации и для управления нашим приложением будет использоваться один из браузеров, доступных в среде Windows и Linux. Браузеры Netscape и Mozilla реализованы для обеих платформ. В настоящее время компания Red Hat включает браузер Mozilla в состав своего дистрибутива Linux, прилагая к нему утилиту для переноса web-закладок из Netscape в Mozilla. Это говорит о том, что браузер Mozilla фактически становится стандартным браузером для Red Hat Linux. Мы не собираемся использовать технологии, которые поддерживаются только браузером Internet Explorer, так как мы должны обеспечить полноценную кроссплатформенную функциональность любого разрабатываемого нами web-приложения. К технологиям, которые поддерживаются только браузером Internet Explorer, относятся поддержка компонентов ActiveX, интеграция с графическим рабочим столом Windows и высокоуровневая поддержка XML. Первые две технологии специфичны для среды Windows, они не поддерживаются в среде Linux. Можно считать, что, отказываясь от их использования, мы ничего не теряем, так как они основаны на механизмах Windows, которые не существуют в Linux.

Однако отказ от высокоуровневой поддержки XML — это в некотором роде потеря для нас. Было бы неплохо, если бы в браузере Netscape мы могли бы преобразовывать XML в HTML так, как мы можем делать это в браузере Internet Explorer. Если бы в Netscape была бы встроена аналогичная стандартная поддержка XML, мы могли бы отображать полученные с сервера XML-данные в соответствии с набором стилевых правил. К сожалению, если мы хотим выполнить такое отображение вне зависимости от того, работает ли наш клиент с Windows XP или с Red Hat Linux 7.1, мы вынуждены выполнять необходимое преобразование XML на стороне сервера. Это значит, что на сервер накладывается дополнительная нагрузка. Данная технология все чаще используется вместо ASP (Active Server Pages) или JSP (Java Server Pages). Подробнее о JSP рассказывается несколько далее.

Мы предполагаем, что третьим звеном является звено данных. Хранение данных осуществляется с использованием DBMS. Бизнес-логика может обращаться к DBMS либо напрямую, либо через промежуточную программную систему, такую как, например, система обмена сообщениями IBM MQSeries. Если для

реализации среднего звена используется Java, то для доступа к звену данных могут использоваться JSM и JDBC.

Говоря о web-программировании, больше всего внимания уделяют, как правило, среднему звену. Среднее звено — это посредник между клиентом и данными. Проблема программирования среднего звена заключается в том, что вы должны предоставить огромному количеству пользователей возможность работать с единственным посредником и при этом обеспечить приемлемо короткое время ответа на запросы. В данном случае пользователь является самой несовершенной и самой неудобной частью дизайна. По сравнению с компьютерами мы, люди, выполняем те или иные действия крайне медленно. Добавьте к этому то, что большую часть времени мы просто смотрим в экран или, хуже того, разговариваем друг с другом. Проблема сводится к обеспечению общего доступа к приложению и поддержке состояния пользовательского сеанса работы с приложением при условии, что соединение с клиентом осуществляется с использованием протокола HTTP, который не поддерживает передачу информации о состоянии сеанса.

Давайте рассмотрим некоторые технологии среднего звена, которые можно использовать как в Windows XP, так и в Linux при разработке приложений компании Elsolutions.

CGI, Common Gateway Interface

Технология CGI (Common Gateway Interface) является корнем фактически всего web-программирования. В мире web-технологий CGI является общим знаменателем, на основе которого строятся многие решения и механизмы. В наши дни CGI по-прежнему активно используется в комплексе с сопутствующими технологиями, позволяющими минимизировать существенные ограничения производительности, связанные с использованием CGI в чистом виде. Причиной низкой производительности является то обстоятельство, что код CGI запускается в виде отдельного процесса, который создается в ответ на каждый из клиентских запросов и уничтожается сразу же, как только обработка запроса завершена. Иными словами, если к системе обращаются 100 клиентов, система вынуждена вначале создать 100 процессов, а затем уничтожить их. Современные технологии позволяют заранее создавать несколько экземпляров CGI-процесса, постоянно находящихся в памяти и ожидающих поступления клиентских запросов. В этом случае при поступлении запроса не требуется создавать новый процесс — для обслуживания клиента используется процесс, извлеченный из пула заранее созданных экземпляров CGI-процесса. Примером подобной технологии является Fast CGI (FCGI). Благодаря использованию подобного подхода вы можете увеличить производительность вашего web-приложения на пару порядков.

Экземпляры процесса CGI изолированы друг от друга, и это обстоятельство является как преимуществом, так и недостатком. С одной стороны, каждый клиентский запрос обслуживается с использованием своего собственного только что созданного процесса. Когда обслуживание запроса завершается, процесс

уничтожается. Это означает, что выполняется полная очистка, а любые утечки памяти блокируются. С другой стороны создание процесса связано с затратами, которые существенно снижают производительность системы, если количество клиентских запросов достаточно велико.

Очевидным преимуществом CGI является простота внедрения. Вместо статических страниц в каталоге web-сервера размещаются исполняемые файлы. Чтобы установить CGI-программу, достаточно просто скопировать ее в этот каталог. Web-сервер может отличать CGI-программы от обычных статических файлов благодаря наличию специального файлового расширения. Например, мы можем сообщить web-серверу, что все файлы с расширением .cgi являются исполняемыми файлами и их требуется запускать на исполнение, а не передавать клиенту. Одним из таких файлов может быть, например, файл order.cgi.

Составные части CGI используются в качестве базиса при разработке более свежих технологий. CGI использует стандартный набор переменных окружения для передачи параметров из HTPP в приложение. Переменные окружения выбраны потому, что они присутствуют фактически на любой платформе. Протокол HTTP передает от клиента к серверу информацию, содержащуюся в URL. Данный метод кодирования используется другими технологиями, так как они тоже используют HTTP. Например, параметры передаются в виде пар «имя=значение». Между именем параметра и его значением ставится символ равенства (=). Пары отделяются друг от друга символом амперсанда (&). Пробелы заменяются символом «плюс» (+). Специальные символы, не являющиеся символами-разделителями, кодируются символом процента (%), за которым следует двухзначный шестнадцатеричный код. Любая программа CGI, равно как и любая другая web-программа, принимающая или формирующая поток данных в формате HTTP, должна обладать способностью кодировать и декодировать значения в соответствии с данной схемой.

Для создания CGI-приложения можно использовать язык C или любой поддерживаемый web-сервером язык сценариев. Некоторые языки более удобны для создания CGI-приложений, чем другие, — это определяется такими факторами, как простота кодирования HTTP-потока или производительность. Зачастую приходится иметь дело с конфликтующими факторами. Например, язык C позволяет создавать чрезвычайно быстрые CGI-программы, однако, программируя CGI-программу на C, приходится решать массу проблем, таких как неудобство кодирования URL. К счастью помочь в данной ситуации может библиотека CGI.

Исторически наиболее популярным языком, используемым для создания CGI-приложений, является Perl. По богатству и разнообразию возможностей библиотека языка Perl может сравниться разве что с библиотекой Java. В библиотеке Perl можно найти уже реализованный код для решения фактически любой задачи. Проблема Perl состоит в том, что это интерпретируемый язык. CGI-программа, написанная на Perl, интерпретируется каждый раз при поступлении клиентского запроса. Очевидно, что это негативно влияет на производительность. Хорошим решением этой проблемы является Mod Perl, о котором рассказывается далее. Ранее мы уже рассматривали Perl в главе 4.

Технология CGI в компании Elsolutions будет использоваться для реализации сетевых приложений, не связанных с обработкой значительного количества клиентских запросов. Например, это могут быть административные приложения. Для создания приложений CGI будут использоваться языки Perl и PHP.

Mod Perl

Mod Perl — это модуль, загруженный в web-сервер Apache и его производные. Для обработки клиентских запросов в сервере Apache используется дизайн типа «цепочка команд» (chain-of-command). Эта цепочка состоит из загружаемых модулей. Модуль Mod Perl написан на C, однако после загрузки он позволяет подгружать другие модули, написанные на Perl. Модуль Mod Perl позволяет регистрировать программы Perl и поддерживает их исполнение в границах одного процесса, без необходимости перезагрузки и перекомпиляции. Да, при загрузке программы Perl компилируются в промежуточную форму исполняемого кода. Таким образом Mod Perl позволяет ускорить работу CGI-системы на два порядка.

Почему мы говорим об Apache? Сервер Apache реализован фактически одинаково и в Linux, и в Windows, поэтому он идеально подходит для использования в смешанных средах Windows/Linux. Незначительные различия между версиями Apache for Windows и Apache for Linux являются следствием того, что в Windows и Linux процессы и программные потоки трактуются несколько поразному. В главе 19 о сервере Apache будет рассказано подробнее. Код CGI-приложения Mod Perl будет исполняться одинаково на обоих платформах с условием, что этот код не зависит напрямую от особенностей той или иной платформы. Избежать этого несложно, так как модули Perl, работа которых напрямую зависит от платформы, в своих именах идентифицируют ту или иную технологию, например, Win32.

Mod Perl будет использоваться в компании Elsolutions для построения административных приложений интранет или приложений-прототипов, на которых будут отрабатываться различные пробные концепции.

Web-фильтры

Серверы Web используют дизайн типа «цепочка команд» (chain-of-command). Цепочка состоит из подгружаемых модулей, которые выполняют обработку потока данных. Каждый модуль, входящий в состав цепочки, выполняет фильтрацию направленных ему данных. Примером web-фильтра является Mod Perl.

Интерфейсы API, специфичные для сервера

Серверы Web различных поставщиков, как правило, обладают специальными программными интерфейсами (API), предназначенными для добавления в сервер дополнительных расширений и фильтров. Все эти расширения работают в рамках серверного процесса. Эти API можно использовать для разработки трехзвенного web-приложения, однако такой подход не является общеприня-

тым. Программный интерфейс web-сервера прежде всего служит для подключения к серверу встраиваемых программных модулей, расширяющих его возможности в соответствии с новыми стандартами. Например, встраиваемый модуль может добавлять в web-сервер поддержку нового языка сценариев или обеспечивать дополнительный механизм защиты. Модуль Mod Perl — один из таких модулей, который позволяет web-серверу Apache напрямую исполнять сценарии Perl. Модуль Mod Perl взаимодействует с web-сервером Apache через специальный программный интерфейс.

Web-сервер Microsoft Information Services (IIS) обладает собственным API, основанным на DLL, этот интерфейс называется ISAPI (Internet Server API). Web-сервер компании Netscape также обладает своим API под названием NCAPI. Мы полагаем, что API web-сервера должен использоваться для расширения возможностей web-сервера, но не для создания трехзвенных web-приложений. Работая над созданием приложений в компании Elsolutions, мы будем изучать возможности использования каждого из API тогда, когда в этом возникнет необходимость.

SSI, Server Side Includes

В наше время все полноценные web-серверы поддерживают технологию SSI (Server Side Includes). SSI — это формальный набор тегов, который обрабатывается web-сервером по мере того как содержимое извлекается из запрашиваемого клиентом файла для дальнейшей передачи через сеть клиенту. Сервер анализирует теги обслуживаемого HTML-документа, некоторые из этих тегов обрабатываются определенным образом и удаляются из результирующего HTML-кода. SSI часто используется для добавления в различные web-документы общих заголовков, элементов оформления, отметок времени и т. п. В результирующем HTML-коде, передаваемом через сеть клиенту, не остается ни одного тега SSI. Типом MIME для содержимого HTML, обрабатываемого с использованием механизма SSI, является тип SHTML. Появление SSI стало предвестником разработки таких технологий, как PHP, ASP и JSP.

PHP

Сокращение PHP означает PHP Hypertext Processing (обработка гипертекста с использованием PHP). Звучит как циклическое определение, не правда ли? Изначально сокращение PHP расшифровывалось как Personal Home Pages. По всей видимости, эту технологию разработал человек, который хотел повысить свою продуктивность в разработке web-приложений. Язык PHP стремительно развился в популярный подпольный стиль разработки web-приложений. Возможно, немногие пользователи слышали об этой технологии, и все же она лежит в основе очень многих web-узлов. Например, по адресу http://www.springmail.com/ располагается служба электронной почты, использующая web-интерфейс. Эта служба целиком и полностью основана на технологии PHP. Разработчики узла перешли на PHP, отказавшись от аналогичной технологии Microsoft ASP, которая эксплуатировалась данным web-узлом в течение шести месяцев. О PHP

рассказывается также в главе 4. Последняя версия PHP4 позволяет компилировать сценарий таким образом, чтобы его можно было повторно использовать внутри web-сервера, без необходимости повторной интерпретации. Сценарий PHP можно запустить независимо, а можно выполнить с использованием специального модуля Apache с названием Mod PHP. Сценарий PHP будет одинаково хорошо работать как в среде Windows, так и в среде Linux. Как правило, для этого используется Apache.

Сценарий PHP можно встроить в код HTML, подобно тому как это происходит при использовании SSI. В процессе обработки такого HTML-файла код PHP обрабатывается сервером и удаляется из результирующего web-документа, возвращаемого клиенту. Вместо сценария PHP в результирующий web-документ как правило добавляются данные, являющиеся результатом работы сценария. Когда клиент получает результирующий web-документ, он видит только статический HTML-код.

Язык PHP4 поддерживает сохранение информации о сеансе, что позволяет программисту помещать в концептуальный сеанс ключевые значения для того чтобы в дальнейшем извлекать их при обработке последующих запросов клиента. Благодаря этому существенно упрощается разработка приложений, в которых используются множественные циклы «запрос — ответ» между клиентом и сервером. Примером подобного приложения является электронный магазин, где каждому пользователю выделяется виртуальная корзина, в которую он может складывать товары. Для хранения информации о состоянии сеанса используются механизмы HTTP cookies и перезапись URL.

Язык PHP будет активно использоваться в компании Elsolutions для внутренних проектов.

ASP, Active Server Pages

Принцип работы Microsoft ASP точно такой же, как и у PHP. Технология ASP поддерживается web-сервером Microsoft IIS, однако существуют также реализации независимых разработчиков, предназначенные для платформы Linux. Существует даже реализация, написанная на Perl. Технология ASP зачастую используется для разработки сценариев работы с компонентами COM, которые существуют только в среде Windows. На наш взгляд технология ASP плохо подходит для компании Elsolutions, так как основным критерием выбора технологий в данной компании является взаимосовместимость и взаимодействие между Windows XP и Red Hat Linux 7.1. Помимо Perl и PHP существует множество других альтернатив.

Сервлеты Java

Прежде чем приступить к обсуждению JSP в качестве альтернативы ASP и PHP, давайте рассмотрим технологию сервлетов Java (Java HTTPD Servlet). Если говорить простым языком, технология сервлетов HTTP Java — это пул экземпляров CGI-программы, написанной на Java. Однако у этой технологии существуют особенности, выгодно отличающие ее от CGI. Программа CGI, написанная

на Java, обладала бы низкой производительностью, так как для каждого пользовательского запроса приходилось бы запускать и останавливать виртуальную машину JVM. В отличие от программы CGI сервлет запускается с использованием JVM и продолжает оставаться в памяти даже после того как пользовательский запрос обработан. Данный, уже существующий в памяти, сервлет может использоваться для обработки последующих пользовательских запросов. Сервлет не поддерживает сохранение информации о сеансе, если только программист не обрабатывает cookies самостоятельно. Для сервлета доступен объект сеанса HTTP. Информацию о сеансе можно хранить в cookies, либо при помощи перезаписи URL (в зависимости от типа сервера приложений, на котором размещается пул сервлетов).

Как правило, сервлет исполняется на сервере приложений. Web-сервер занимается обработкой статического содержимого и программ, не являющихся исполняемым кодом Java. Запросы к сервлетам могут перенаправляться серверу приложений Java. Сервер приложений Java может быть отдельным локальным или удаленным процессом. Часто web-сервер использует протокол с названием AJP, который применяется через сокетное соединение TCP для взаимодействия с сервером приложений. Фермы web-серверов с распределением нагрузки могут служить интерфейсом для доступа клиентов к сервлетам, работающим на серверах приложений Java. Серверы приложений могут быть реализованы в виде серверных кластеров. Сервлеты — это часть web-приложения, размещенная на сервере приложений. Различные серверы могут использоваться для обеспечения функционирования различных web-приложений. Отдельное web-приложение часто называют *зоной сервлетов* (servlet zone). Web-сервер может располагаться между брандмауэрами в так называемой демилитаризованной зоне (DMZ), в то время как сервер приложений может располагаться во внутренней сети, под защитой внутреннего брандмауэра. Таким образом формируется двухуровневая защита от злонамеренных действий злоумышленника.

Сервлет может располагаться в любом месте, где есть Java. Предполагается, что на платформе, которую вы выбрали в качестве сервера приложений, существует реализация Java. В данной области существует множество разнообразных решений. Сервер приложений IBM WebSphere реализован как для Linux, так и для Windows. Проект Apache Jakarta включает в себя реализацию сервера приложений для сервлетов и JSP под названием Tomcat. Этот продукт поставляется с открытым исходным кодом. Загрузить Tomcat можно по адресу http://www.apache.org/, где следует войти в раздел Jakarta (общее имя для нескольких Java-проектов). Данный сервер можно использовать со своим собственным web-сервером, а можно подключить к таким web-серверам, как Apache, Netscape или IIS. Сервер Tomcat может работать как в Windows, так и в Linux, он отлично подходит для ознакомления с технологиями сервлетов и JSP.

Если вы используете сервлеты Java без каких-либо сопутствующих технологий, вы обнаруживаете, что с их помощью не удается четко отделить модель от представления (model-view). Результирующий код HTML должен быть выведен в поток вывода сервлета. Таким образом, код GUI тесно связывается с про-

цедурной моделью приложения. Для решения этой проблемы удобно использовать JSP.

JSP, Java Server Pages

Почему мы не рассмотрели JSP до того, как приступить к изучению сервлетов? Дело в том, что JSP — это всего лишь еще один способ написания сервлета. JSP — это язык сценариев, включающий в себя синтаксис Java и набор специальных тегов для включения Java-кода в состав HTML. Java-код, обозначенный специальными тегами, входит в состав потока даных HTML и обрабатывается сервером подобно тому, как это происходит с PHP и ASP. Чтобы подчеркнуть разницу между JSP и JavaScript, сценарии, написанные на JavaScript, называют *скриптлетами* (scriptlet). Отличительная черта JSP состоит в том, что, в отличие от JavaScript, вся страница JSP компилируется в сервлет Java. Полученный сервлет исполняется точно так же, как и любой другой сервлет Java. Таким образом, сценарии JSP обладают существенно более высокой производительностью по сравнению с PHP и ASP. В последнее время сценарии, написанные на JSP, также часто называют скриптлетами JSP.

Зачем нужен такой необычный метод создания сервлетов? Дело в том, что человек, занимающийся разработкой GUI, может использовать для создания пользовательского web-интерфейса удобный специализированный инструмент, такой как IBM WebSphere Studio. Для PHP и ASP характерна одна и та же проблема: при создании сложных приложений размер страниц стремительно увеличивается, в результате вы получаете в свое распоряжение чудовищную смесь исполняемого кода и языка разметки. Такие приложения сложно тестировать и сопровождать. Если для создания web-приложения использовать только JSP, мы можем столкнуться с такой же проблемой. Мало того, сам по себе сценарий JSP плохо приспособлен для отделения модели от представления (model-view) — в данном случае эта проблема также актуальна. В чем же состоит решение?

Чтобы решить все проблемы, необходимо изменить подход к проектированию нашего приложения. Сервлет может передавать запросы другому сервлету. В качестве целевого сервлета можно использовать сценарий JSP. Код JSP обладает возможностью передавать и принимать в качестве аргументов компоненты JavaBean, для этого используется специальный тег `jsp:usebean`. Компонент JavaBean — это простой черный ящик, обладающий интерфейсом, устройство которого можно узнать динамически.

Исходя из всего вышесказанного сформулируем схему нашего приложения. Если говорить в терминах модели «модель — представление — контроллер» (Model — View — Controller, MVC), контроллер и модель нашего приложения реализованы в виде обычного сервлета (или набора сервлетов) Java. Динамическое представление информации (View) реализовано в виде сценария JSP, который тоже фактически является сервлетом. Информация передается от сервлета сценарию JSP в виде компонента JavaBean — для этого можно использовать либо объект запроса, либо объект сеанса. Объект запроса уничтожается после того как завершается обслуживание клиентского запроса. Объект сеанса существует до тех пор, пока не истечет тайм-аут либо не будет произведено его явное

уничтожение. Для указания типа объекта JavaBean и области его видимости используется тег jsp:usebean. Свойства компонента используются для настройки значений полей формы HTML или генерируются скриптлетами JSP.

Что заносится в компонент JavaBean? Как правило, приложение извлекает данные из хранилища данных, то есть третьего звена. В компонент JavaBean можно занести результирующий набор (result set), полученный с использованием JDBC. Получив этот компонент, сценарий JSP может отобразить эти данные в составе таблицы HTML. Часть данных может быть обновлена при помощи HTML-формы. Затем данные могут быть переданы обратно сценарию JSP, который может разместить данные в компоненте JavaBean и передать этот компонент обратно сервлету Java, выполняющему функции контроллера. Получив данные, сервлет Java обновляет базу данных на основе информации, полученной из свойств сервлета.

Может показаться, что приложение Hello World, написанное с использованием данного подхода, окажется чрезвычайно громоздким и неповоротливым, однако на самом деле объем дополнительных затрат фиксирован, — он фактически не зависит от масштаба приложения. Если вы делите приложение на JSP и сервлеты Java, обращение к DBMS и графический пользовательский интерфейс разрабатываются значительно проще — именно в этом и заключается преимущество трехзвенного программирования. Полученную в результате программу проще кодировать, тестировать, расширять, сопровождать и понимать. Кроме того, она хорошо масштабируется в случае если требуется обеспечить большую производительность.

Мы предполагаем, что схема «сервлет Java/сценарий JSP» будет использоваться в компании Elsolutions для построения web-приложений корпоративного уровня. Мы будем использовать эту технологию как для создания сетевых приложений на заказ, так и для разработки внутренних высокопроизводительных сетевых приложений.

Демонстрация web-технологий с использованием портируемой тестовой программы

Давайте посмотрим, каким образом каждая из рассмотренных технологий может функционировать в рабочей среде Elsolutions. Для демонстрации мы напишем простое приложение под названием Echo (эхо), которое отображает на экране переданное ему в качестве параметра значение и добавляет к этому информацию о дате и времени. Это приложение будет реализовано шесть раз — с использованием шести различных технологий. Каждая из технологий, за исключением программы, написанной на C, подразумевает бинарную совместимость исполняемых файлов для Windows и Linux. Это означает, что исполняемый файл без изменений и перекомпиляции может быть перенесен из Windows в Linux и обратно. На рис. 6.9 показана HTML-панель, при помощи которой вы можете обратиться к любому из шести этих приложений (соответствующий HTML-файл

называется echo.html). При щелчке на одной из кнопок будет запущено приложение, реализованное с использованием указанной технологии. Это приложение исполняется и в качестве результата своей работы возвращает HTML-код, который содержит дату, время, текст, переданный в качестве параметра, а также гиперссылку, указывающую обратно, на основную HTML-панель.

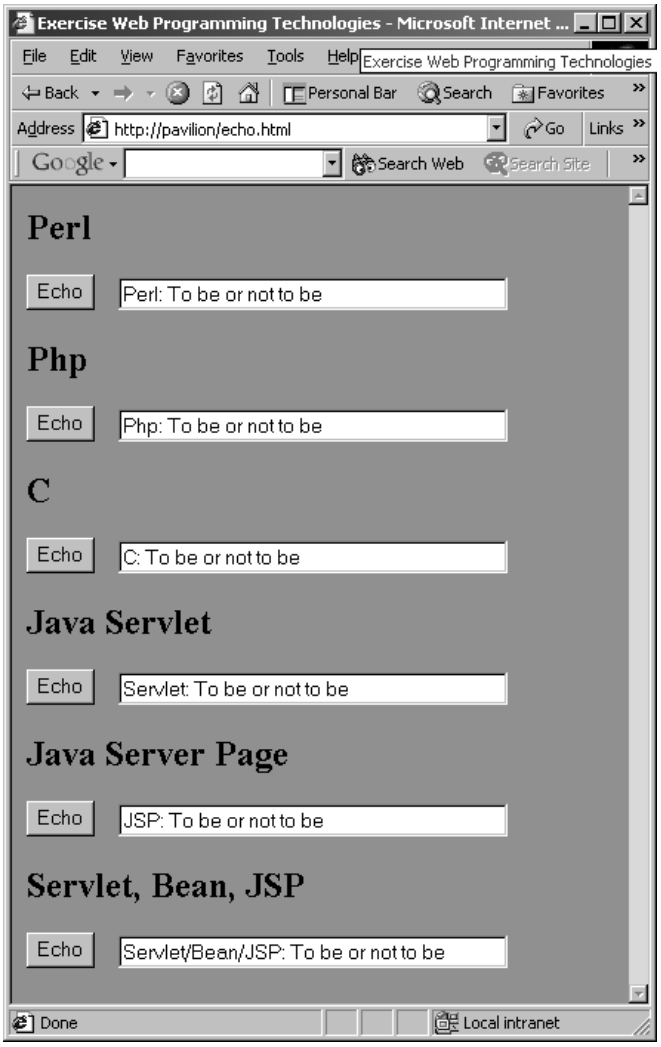

Рис. 6.9. Главная HTML-панель нашего демонстрационного приложения

Для запуска этого тестового приложения как в Windows, так и в Linux будет использоваться web-сервер Apache. В качестве сервера приложений Java будет использоваться Tomcat, соединенный с Apache. Продукт Tomcat легко устанавливается и легко соединяется с Apache. Все, что нужно сделать, это добавить

выражение Include, указывающее на конфигурацию Tomcat (при этом, конечно же, необходимо добавить в Apache модуль Mod Jserv, благодаря чему Apache получает возможность исполнять модули Java).

Таким же образом модуль Mod PHP используется для того чтобы добавить в Apache поддержку PHP. В нашем сервере Apache установлен также модуль Mod Perl, однако для запуска сценария Perl этот модуль не используется, — вместо этого мы запускаем тестовую программу Perl в виде обычного CGI-приложения.

Далее приводится HTML-код главной панели запуска тестовых приложений (содержимое файла echo.html). Этот HTML-документ размещается в каталоге HTML-документов сервера Apache в среде Linux и Windows.

```
<html>
<head>
<title>Exercise Web Programming Technologies</title>
</head>
<body bgcolor="#909090">
<h2>Perl</h2>
<form method="post" action="/cgi-bin/echo.pl">
<input type="submit" name="submit" value="Echo">
  
<input type="text" name"echoperl" size="40"
value="Perl: To be or not to be"><br>
</form>
<h2>Php</h2>
<form method="post" action="/echo.php">
<input type="submit" name="submit" value="Echo">
  
<input type="text" name"echophp" size="40"
value="Php: To be or not to be"><br>
</form>
<h2>C</h2>
<form method="post" action="cgi-bin/echo">
<input type="submit" name="submit" value="Echo">
  
<input type="text" name"echoc" size="40"
value="C: To be or not to be"><br>
</form>
<h2>Java Servlet</h2>
<form method="post" action="/book/servlet/book.interop.Echo1">
<input type="submit" name="submit" value="Echo">
  
<input type="text" name"echojava" size="40"
value="Servlet: To be or not to be"><br>
</form>
<h2>Java Server Page</h2>
<form method="post" action="/book/jsp/echo.jsp">
<input type="submit" name="submit" value="Echo">
  
<input type="text" name"echojava" size="40"
value="JSP: To be or not to be"><br>
</form>
<h2>Servlet, Bean, JSP</h2>
<form method="post" action="/book/servlet/book.interop.Echo2">
<input type="submit" name="submit" value="Echo">
  
<input type="text" name"echojava" size="40"
```

```
value="Servlet/Bean/JSP: To be or not to be"><br>
</form>
</body>
</html>
```

Вывод приложения, написанного на Perl, показан на рис. 6.10.

Рис. 6.10. Вывод программы Echo, написанной на Perl

Исходный код программы на языке Perl показан далее. Соответствующий исполняемый файл располагается в каталоге cgi-bin сервера Apache (вне зависимости от платформы). Обратите внимание на то, что мы используем Perl-модуль с именем CGI. Можно заметить, что приложение получилось очень простым. Основной код, предназначенный для создания HTML-разметки, содержится в модуле CGI. Модуль CGI написан, естественно, на Perl, поэтому его код будет одинаково хорошо исполняться как в Linux, так и в Windows. Естественно, в среде Windows вам потребуется установить реализацию Perl (которую можно получить бесплатно).

```
#! /usr/bin/perl -w
use CGI qw(-no_debug :standard);
use strict;
my $q = new CGI;
my $tm = localtime();
my $echoparam = $q->param("echoperl");

print $q->header({-BGCOLOR=>'#909090'}),
      $q->start_html({
                -title=>"Perl Echo"
                -BGCOLOR=>"#b0b0b0"},
                "Echo Implemented in Perl\n"),
```

```
$q->h1("Perl of Wisdom\n"),
$q->p("Hello, World! It is now $tm.\n"),
$q->p("You entered: "),
$q->strong("'$echoparam'"),
$q->p(),
$q->a({href=>"/echo.html"}, "| Home |"),
$q->hr(),
$q->end_html;
```

На рис. 6.11 показана версия программы Echo, написанная на PHP.

Рис. 6.11. Программа Echo, написанная на PHP

Исходный код приложения, написанный на PHP, показан далее. Он отличается от обычной HTML-страницы всего несколькими PHP-выражениями. Данный файл необходимо разместить в каталоге html сервера Apache. Чтобы запустить этот сценарий в среде Windows, вам потребуются дистрибутив PHP и модуль Mod PHP для Apache. Дистрибутив Red Hat Linux 7.1 обладает всем необходимым для запуска сценариев Perl и PHP. Если на вашей платформе есть все необходимое, данный сценарий без каких-либо модификаций будет успешно выполняться как в среде Windows, так и в среде Linux.

```
<html>
<head><title>Php Echo Demo</title></head>
<body bgcolor="#b0b0b0">
<h1>PHP Echo Panel</h1>
<p>Hello, World! It is now
<?php echo  (date("D M j H:i:s Y")); ?>
<p>You entered:</p>
<strong>'<?php echo ($echophp)?>'</strong>
```

```
<p><a href="/echo.html">| Home |</a>
<hr>
</body>
</html>
```

Вывод приложения Echo, написанного на C, показан на рис. 6.12.

Рис. 6.12. Вывод программы Echo, написанной на C

Соответствующий код C (на самом деле для удобства используется синтаксис CPP) получился настолько большим, что мы не стали приводить его в данной книге. Результирующий код HTML формируется многочисленными вызовами printf, выводящими информацию в стандартный поток вывода stdout программы. Большая часть кода выполняет декодирование URL, извлечение из URL параметров, получение динамического списка пар «имя=значение». Другими словами, разрабатывая CGI-приложение на C, мы вынуждены обрабатывать текстовую информацию на достаточно низком уровне. К счастью, код подобных процедур может использоваться любыми CGI-приложениями, написанными на C.

В мире приложений среднего звена язык C не столь популярен, как Perl или Java. Язык C обладает низкоуровневым синтаксисом; кроме того, управление памятью в C связано с проблемами. Если программа используется в качестве традиционного CGI-приложения, то есть для обслуживания каждого запроса запускается новый экземпляр процесса, то управление памятью может быть достаточно неряшливым, так как после завершения обслуживания запроса процесс все равно удаляется из памяти. Однако в этом случае общая производительность системы будет низкой из-за затрат, связанных с созданием новых экземпляров процесса. Чтобы на два порядка увеличить производительность обычно-

го CGI-приложения, требуется использовать технологию, подобную FCGI. Однако при использовании FCGI (и ей подобных механизмов) программа C сохраняется в памяти и используется для обработки множества клиентских запросов. Каждый новый запрос вызывает повторное исполнение одного и того же кода. В этом случае следует обеспечить чрезвычайно аккуратное освобождение всей используемой памяти после завершения обработки запроса. Утечки памяти могут негативно повлиять на производительность или, того хуже, полностью разрушить систему.

Давайте теперь перейдем к рассмотрению решений, основанных на Java. Вывод версии приложения Echo, реализованной в виде сервлета Java, показан на рис. 6.13.

Рис. 6.13. Вывод программы Echo, реализованной в виде сервлета Java

Далее приводится код метода doPost, в рамках которого формируется этот вывод. Используемая нами форма подразумевает обработку метода HTTP POST. Метод doGet просто возвращает сообщение об ошибке. Оба этих метода являются переопределениями методов класса HttpServlet. Если вы хотите выполнить однократную инициализацию в момент загрузки сервлета, вы можете переопределить метод init. Сервлет загружается либо в момент запуска сервера приложений, либо в момент первого использования. Мы можем приказать серверу приложений перезагружать сервлет в случае если его код меняется, однако в этом случае сервер будет анализировать дату создания сервлета каждый раз, когда к сервлету происходит обращение, а это означает, что возникнут дополнительные затраты времени.

```
/**
 * Обработка поступающих запросов HTTP POST
 * Отображение даты и времени.
 * Отображение значения параметра запроса
 *
 * @param request  - это объект, который инкапсулирует в себе
 *                   запрос к сервлету
 * @param response - это объект, который инкапсулирует в себе
 *                   ответ от сервлета
 */
public void doPost(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response)
         throws javax.servlet.ServletException, java.io.IOException {

    StringBuffer buf = new StringBuffer(400);

    buf.append("<html><head><title>Servlet Echo Demo</title></head>
        <body bgcolor=\"#b0b0b0\">\n);
    buf.append("<h1>Java Servlet Echo Panel</h1><p>
        Hello, World! It is now \n");
    // Вставляем день, месяц, число, час, минуту, секунду и год.
    // Форматируем текущее время.
    SimpleDateFormat formatter
         new SimpleDateFormat ("EEE MMM dd kk:mm:ss yyyy");
    Date currentTime_1 = new Date();
    String dateString = formatter.format(currentTime_1);
    buf.append(dateString).append("\n");

    buf.append("<p>You entered:</p><strong>'\n")ж

    // Вставляем переданный эхо-параметр
    String value = request.getParameter("echojava");
    if (value != null);
       buf.append(value);
    else
       buf.append("ERROR: *** Nothing entered ***");

    buf.append( "'</strong><p>
          <a href=\"http://pavilion/echo.html\">
          | Home |</a>\n");
    buf.append("<hr></body></html>\n");

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.print(buf.toString());
}
```

Мы можем сконфигурировать сервлет таким образом, чтобы его запуск осуществлялся в случае если происходит обращение к определенному имени в рамках нашего web-приложения. Передача инициализационных параметров осуществляется при помощи конфигурационного сценария (как правило, написанного на XML). Другой способ запуска сервлета подразумевает обращение к каталогу web-приложения, за именем которого указывается суффикс /servlet, а затем — имя класса сервлета. В нашем случае сервлет запускается другим сервлетом, который обозначается английским термином *invoker* (вызывающий). Этот сервлет обнаруживается системой, так как сконфигурирован соответствующим образом.

В данном демонстрационном приложении мы используем вызывающий сервлет для запуска других сервлетов. Наше web-приложение называется book. Не забывайте о том, что наш сервлет может работать как в Windows, так и в Linux без перекомпиляции, однако при этом на обеих платформах должны быть установлены JVM одной и той же версии.

Вывод приложения Echo, написанного на JSP, показан на рис. 6.14.

Рис. 6.14. Вывод программы Echo, реализованной в виде страницы JSP

Далее показан исходный код JSP. Данная страница JSP размещена в рамках web-приложения book, в каталоге jsp сервера приложений Tomcat. Данный сценарий JSP исполняется одинаково как в Windows, так и в Linux.

```
<html>
<head><title>JSP Echo Demo</title>
<%@ page import="java.util.*"%>
</head>
<body bgcolor="#b0b0b0">
<h1>JSP Echo Panel</h1>
<p>Hello, World!
<%= "It is now " + new Date() %>
<p>You entered:</p>
<strong>'<%= request.getParameter("echojava") %>'</strong>
<p><a href = "/echo.html">| Home |</a>
<hr>
</body>
</html>
```

А теперь приступим к рассмотрению методики, которая будет использоваться нами в качестве основной при разработке портируемых, масштабируемых

и расширяемых web-приложений Java. Условная схема показана на рис. 6.15. Конечно же, при реализации программы Echo мы не используем звено данных так, как показано на этом рисунке, однако здесь видно, каким образом передаются данные в процессе обработки клиентского запроса. Для хранения данных в процессе обработки запроса (или даже в течение всего рабочего сеанса) мы решаем использовать JavaBean. Экземпляр компонента JavaBean вставляется в принадлежащий сервлету объект запроса (или, соответственно, в объект сеанса). Компонент JavaBean идентифицируется при помощи метки jsp:usebean, предоставляя этому компоненту область сеанса или область сессии в рамках сценария JSP.

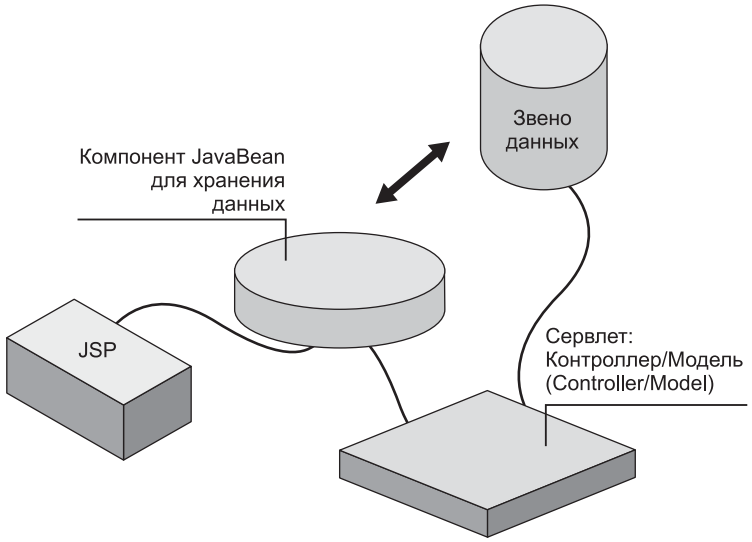

Рис. 6.15. Схема MVC (Model — View — Controller), реализованная средствами Java

Реальное приложение для обращения к хранилищу данных из сервлета будет использовать интерфейс JDBC. Сервлет в свою очередь может быть разделен на несколько функциональных частей. Одна из частей отвечает за передачу презентационных данных сценарию JSP, остальные части являются объектами, в виде которых представлена внутренняя бизнес-логика приложения. Также должен существовать уровень отображения реляционных данных во внутренние данные объектов, который должен изолировать объекты приложения от специфики механизма долговременного хранения данных. Однако все эти замечания относятся к реальному web-приложению, работающему с реальной базой данных. На самом деле мы отвлеклись от конкретного рассматриваемого нами примера. Рассматриваемое нами приложение Echo имеет значительно более простую структуру.

Сервлет не содержит в себе никакой презентационной логики за исключением, может быть, отображения ошибки в HTML. Даже эта функция перекладывается на специальный сценарий JSP или сервлет. Работа основного сервлета заключается в получении компонента JavaBean, извлечении данных из JavaBean,

обработке этих данных и записи данных в JavaBean. Экземпляр компонента JavaBean размещается либо в объекте запроса, либо в объекте сеанса. После этого сервлет передает запрос сценарию JSP. Тег jsp:usebean информирует сервлет о том, в каком месте следует искать экземпляр компонента JavaBean. После этого значения свойств компонента JavaBean применяются для использования при построении динамического содержимого. Свойства компонента могут быть связаны с полями формы; благодаря этому данные, хранящиеся в компоненте JavaBean, будут обновляться в результате заполнения пользователем полей формы и нажатия на кнопку submit. Если компонент располагается в объекте сеанса, он будет использоваться сервлетом в следующем цикле обработки.

Теперь давайте рассмотрим простую реализацию этой схемы, в которой используется только объект запроса. Не забывайте, что у нас нет звена данных, однако доступ к данным может быть добавлен в код в точке, где мы принимаем ввод и возвращаем обратно вывод. Взгляните на рис. 6.16. Скорее всего, эта картинка вам уже порядком надоела, однако это последний вариант реализации нашей несложной программы.

Рис. 6.16. Вывод программы Echo, реализованной в виде страницы JSP

Далее приводится текст метода doPost нашего сервлета, этот метод принимает запрос от страницы HTML. Обратите внимание, что метод не выполняет генерацию HTML. Он создает экземпляр компонента JavaBean, принимает значение входного параметра и помещает его в этот компонент. Информация о дате и времени (timestamp) содержится в компоненте JavaBean в виде вычисляемого свойства, предназначенного только для чтения (read-only). Компонент JavaBean

помещается в объект запроса, после чего управление передается сценарию JSP. Обратите внимание на закомментированную строку, в которой компонент JavaBean помещается в объект сеанса. Это выражение следует использовать в случае если вы хотите сохранить содержимое компонента JavaBean до времени обработки следующего клиентского запроса.

```
/**
 * Обработка поступающих запросов HTTP POST
*
 * @param request  - это объект, который инкапсулирует в себе
 *                   запрос к сервлету
 * @param response - это объект, который инкапсулирует в себе
 *                   ответ от сервлета
 */
public void doPost(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) trows
javax.servlet.ServletException, java.io.IOException {

    try {
        // Извлекаем эхо-параметр из запроса
        String value = request.getParameter("echojava");
        if (value == null)
            value = "ERROR: *** Nothing entered ***";

        // Добавляем эхо-параметр компонент Bean
        EchoBean bean = new book.interop.EchoBean();
        bean.setEchoValue(value);

        // При использовании области видимости сеанса:
        // размещаем bean в объекте сеанса в ключе "echobean"
        /*
        HttpSession session = request.getSession(true);
        session.putValue("echobean", bean);
        */

        // При использовании области видимости запроса:
        // размещаем bean в объекте запроса в ключе "echobean"
        request.setAttribute("echobean",bean)   ;

        // Передаем запрос сценарию JSP
        javax.servlet.RequestDispatcher dispatcher
            = getServletContext().getRequestDispatcher("/jsp/echo2.jsp");
        dispatcher.forward(request, response);

    } catch(Throwable the Exception) {
        // Отлаживаем неожидаемые исключения
        theException.printStackTrace();
    }
}
```

Далее приводится код компонента JavaBean. Этот компонент обладает всего двумя свойствами: echoValue и timeStamp. Свойство timeStamp можно только читать, кроме того, значение этого свойства не хранится в компоненте, а вычисляется.

```
package book.interop;
import java.text.*;
import java.util.*;
```

```
import java.io.*;

/**
 * Компонент хранения данных для приложения Echo сервлет/JSP
 * Дата создания: (2/27/2001 9:35:50 PM)
 * @author: Louis E. Mauget
 */
public class EchoBean implements Serializable {
    protected java.lang.String echoValue = "";
    private final static long serialVersionUID = 1243021828975292229L;
/**
 * конструктор EchoData
 */
public EchoBean() {
    super();
}
/**
 * Get echoValue
 * Creation date: (2/27/2001 9:37:20 PM)
 * @return java.lang.String
 */
public java.lang.String getEchoValue() {
    return echoValue;
}
/**
 * Get timestamp
 * Creation date: (2/27/2001 9:37:20 PM)
 * @return java.lang.String
 */
public java.lang.String getTimeStamp() {

    // Возвращаем день, месяц, час, минуту, секунду и год.
    // Форматируем текущее время.
    SimpleDateFormat formatter
         = new SimpleDateFormat ("EEE MMM dd kk:mm:ss yyyy");

    Date currentTime_1 = new Date();
    String dateString = formatter.format(currentTime_1);
     return dateString;
}
/**
 * Get echoValue
 * Creation date: (2/27/2001 9:37:20 PM)
 * @param newEchoValue java.lang.String
 */
public void setEchoValue(java.lang.String newEchoValue) {
    echoValue = newEchoValue;
}
}
```

Далее приводится исходный код страницы JSP. Обратите внимание на тег jsp:usebean. Этот тег является описанием компонента JavaBean. Область видимости этого компонента (scope) установлена равной request, то есть в пределах одного запроса. Это связано с тем, что наш сервлет размещает компонент bean в объекте запроса. Это в свою очередь означает, что компонент будет уничтожен после того, как страница будет показана на экране. Это нас вполне устраивает, так как мы не собираемся передавать компонент куда-либо далее. Если бы мы использовали для хранения JavaBean объект сеанса, параметр scope (область ви-

димости) должен был быть равен значению session (сеанс). Еще один вариант — значение page (страница). В этом случае компонент Bean будет существовать только в течение жизни страницы JSP. Существует еще один вариант области видимости — application (приложение). В этом случае компонент JavaBean может быть размещен в объекте приложения. Это означает, что экземпляр компонента будет существовать в течение всего времени жизни web-приложения. Эта возможность используется для хранения глобальных данных, если конечно у вас возникнет такая необходимость.

```
<jsp:useBean id="echobean" scope="request" class=
"book.interop.EchoBean" />
<html>
<head><title>Servlet/Bean/JSP Echo Demo</title>
<%@ page import="java.util.*"%>
</head>
<body bgcolor="#b0b0b0">
<h1>Servlet/Bean/JSP Echo Panel</h1>
<p>You entered:</p>
<strong>'
<jsp:getProperty name="echobean" property="timeStamp" />
<p>You entered:</p>
'</strong>
<p><a href="/echo.html">| Home |</a>
<hr>
</body>
</html>
```

Используемые в web-программировании средства разработки и развертывания

Все рассмотренные только что страницы были вручную написаны с использованием редактора VIM, запущенного в среде Linux в рабочем сеансе telnet. Мы не использовали для создания наших программ какой-либо специализированный инструмент разработки, так как при этом в результирующий документ, как правило, добавляется большой объем дополнительной служебной информации, в результате чего базовый код страницы становится менее понятным.

После того, как рассматриваемые нами страницы заработали в среде Linux, мы перенесли их в среду Windows 2000. Для этой цели использовались web-сервер Apache и сервер приложений Tomcat. В сервер Apache были добавлены модули Mod Perl, Mod PHP и Mod Jserv. В средах Windows XP и Windows 2000 требуется дополнительно устанавливать интерпретаторы Perl и PHP, а в среде Red Hat Linux 7.1 делать этого не требуется, так как все необходимые интерпретаторы уже включены в состав дистрибутива этой ОС.

Для разработки web-приложений можно использовать самые разнообразные средства, начиная от простых текстовых редакторов vi и Notepad и заканчивая комплексными средами разработки, такими как Sun Forte for Java, Semantic Cafe, Borland JBuilder и IBM Visual Age for Java. Не все эти продукты доступны в среде Linux. Компания IBM предлагает одну из предыдущих версий Visual Age for

Java, работающую в среде Linux. Наиболее свежие версии этого продукта доступны для Windows.

Три из шести версий приложения Echo написаны в среде Linux с использованием соединения telnet. Версии, основанные на Java, были разработаны с использованием продукта IBM Visual Age for Java. Полученный код был экспортирован в JAR-файл и перенесен в среду Linux при помощи FTP. Для написания файлов в среде Linux использовался редактор VIM — разновидность VI. Это весьма удобный редактор, так как он поддерживает цветовую разметку исходного кода, написанного на большинстве известных языков программирования. Редактор VIM знаком с множеством языков, начиная с XML и заканчивая PHP. Встроенный механизм проверки правильности синтаксиса — это также приятная особенность, облегчающая работу программиста.

Существует огромное количество средств разработки документов HTML и JSP. Одним из таких инструментов является IBM WebSphere Studio, однако версии этого продукта для Linux не существует. Еще одним популярным инструментом является Microsoft FrontPage, однако этот продукт не позволяет добавлять в разрабатываемый документ теги JSP. То обстоятельство, что некоторый инструмент разработки web-документов может работать только в среде Windows, не препятствует размещению полученных в результате web-документов в среде Linux. Все рассмотренные нами версии программы Echo, за исключением версии, написанной на C, можно без каких-либо модификаций переносить из Linux в Windows. Являющийся исключением код, написанный на C, требует повторной компиляции. Для компиляции C-программы в среде Windows 2000 и Windows XP мы использовали компилятор GCC в комбинации со средствами Cygnus. Это отлично соответствует компилятору GCC в среде Linux, однако с одинаковым успехом мы могли бы использовать и Microsoft Visual C++. Используя в разных средах разные компиляторы, необходимо уделять особое внимание параметрам компиляции и разнообразным специфичным для данного компилятора возможностям, иначе вы рискуете снизить степень портируемости вашей программы.

Для реализации нашей демонстрационной программы мы использовали Java, Perl, PHP и C. Прежде чем перенести полученный код в среду Windows, нам потребовалось заново перекомпилировать только программу C. Остальные версии одинаково корректно работают в любой из сред без каких-либо дополнительных модификаций и процедур. Однако при разработке реальных web-приложений для обеспечения переносимости нам, возможно, пришлось бы некоторым образом модифицировать исходный код. Зачастую это связано с различием в интерфейсах DBMS, однако очень часто подобные параметры можно вынести во внешний файл свойств. Благодаря этому удается избежать жестких определений в рамках исполняемого кода программы.

Заключение

Мы убедились в том, что Java является основным механизмом обеспечения переносимости и взаимодействия между двумя платформами Windows XP и Red

Hat Linux 7.1 в компании Elsolutions. Стандартизированная, кроссплатформенная природа Java привлекает к себе огромное внимание всех, кто работает в области web-программирования. В результате за последнее время для Java было разработано огромное количество программных интерфейсов и библиотек. Многочисленные средства разработки, упрощающие создание приложений Java, дополняют собой общую картину процветания Java. Благодаря всему этому Java является одной из наиболее эффективных технологий для создания и внедрения корпоративных приложений. В последнее время технология XML становится такой же популярной, как и Java. Комбинация двух этих технологий предлагает разработчику богатый арсенал средств обеспечения переносимости и взаимосовместимости, таких как, например, протокол SOAP, который позволяет связывать между собой службы и клиенты вне зависимости от того, где они находятся, в Windows или в Linux.

Мы рассмотрели несколько программных технологий, которые планируется использовать в компании Elsolutions. В этой области существует большое количество вариантов выбора. Мы обсудили возможность использования в Elsolutions нескольких наиболее популярных языков программирования web-приложений и презентационных технологий.

Мы также разработали несколько версий простой web-программы Echo, реализованных на четырех разных языках. Для технологии Java были представлены три различных варианта дизайна web-приложения. Мы пришли к выводу, что в качестве основной схемы при разработке серьезных web-приложений будет использоваться схема JSP/JavaBean/сервлет. Все разработанные нами версии программы Echo можно без изменений переносить между Linux и Windows. Исключением являются CGI-программы, написанные на C, — такие программы требуется заново откомпилировать, так как в Windows и Linux используется разный формат исполняемых файлов. Приложение Echo — это весьма простая программа, которая использовалась нами только для демонстрационных целей.

Реальные приложения, разрабатываемые в компании Elsolutions, могут включать в себя логику, вставленную между кодом получения запроса и кодом отправления ответа на запрос. Код доступа к данным может быть добавлен в приложение в точке, где прочитано значение параметра и где формируется эхо-ответ. Существует несколько DBMS, общих для Windows и Linux. Интерфейсы JDBC и ODBC позволяют работать с базой данных вне зависимости от поставщика и реализации конкретной DBMS.

Мы также кратко рассмотрели разнообразные средства разработки и внедрения. Распределенная природа web-программирования позволяет разрабатывать, компилировать и развертывать прикладные программы удаленно, то есть через сеть. Благодаря этому, в частности, для разработки программы для среды Linux можно использовать средство, которое работает только в среде Windows.